MCF51QE128
MCF51QE64
MCF51QE32
Reference Manual

ColdFire
Microcontrollers

freescale.com

freescale™
semiconductor

MCF51QE128 Series Features

- 32-Bit Version 1 ColdFire® Central Processor Unit (CPU)
  - Up to 50.33-MHz ColdFire CPU from 3.6V to 2.1V, and 20-MHz CPU at 2.1V to 1.8V across temperature range of -40°C to 85°C
  - Provides 0.94 Dhrystone 2.1 MIPS per MHz performance when running from internal RAM (0.76 DMIPS/MHz from flash)
  - Implements Instruction Set Revision C (ISA_C)
  - Support for up to 30 peripheral interrupt requests and seven software interrupts
- On-Chip Memory
  - Flash read/program/erase over full operating voltage and temperature
  - Random-access memory (RAM)
  - Security circuitry to prevent unauthorized access to RAM and flash contents
- Power-Saving Modes
  - Two low power stop modes; reduced power wait mode
  - Peripheral clock enable register can disable clocks to unused modules, reducing currents; allows clocks to remain enabled to specific peripherals in stop3 mode
  - Very low power external oscillator can be used in stop3 mode to provide accurate clock to active peripherals
  - Very low power real time counter for use in run, wait, and stop modes with internal and external clock sources
  - 6 μs typical wake up time from stop modes
- Clock Source Options
  - Oscillator (XOSC) — Loop-control Pierce oscillator; Crystal or ceramic resonator range of 31.25 kHz to 38.4 kHz or 1 MHz to 16 MHz
  - Internal Clock Source (ICS) — FLL controlled by internal or external reference; precision trimming of internal reference allows 0.2% resolution and 2% deviation; supports CPU freq. from 2 to 50.33 MHz
- System Protection
  - Watchdog computer operating properly (COP) reset with option to run from dedicated 1-kHz internal clock source or bus clock
  - Low-voltage detection with reset or interrupt; selectable trip points
  - Illegal opcode and illegal address detection with programmable reset or exception response
  - Flash block protection
- Development Support
  - Single-wire background debug interface
  - 4 PC plus 2 address (optional data) breakpoint registers with programmable 1- or 2-level trigger response
  - 64-entry processor status and debug data trace buffer with programmable start/stop conditions
- ADC — 24-channel, 12-bit resolution; 2.5 μs conversion time; automatic compare function; 1.7 mV/°C temperature sensor; internal bandgap reference channel; operation in stop3; fully functional from 3.6V to 1.8V

- ACMPx — Two analog comparators with selectable interrupt on rising, falling, or either edge of comparator output; compare option to fixed internal bandgap reference voltage; outputs can be optionally routed to TPM module; operation in stop3
- SCIx — Two SCIs with full duplex non-return to zero (NRZ); LIN master extended break generation; LIN slave extended break detection; wake up on active edge
- SPIx— Two serial peripheral interfaces with Full-duplex or single-wire bidirectional; Double-buffered transmit and receive; MSB-first or LSB-first shifting
- IICx — Two IICs with; Up to 100 kbps with maximum bus loading; Multi-master operation; Programmable slave address; Interrupt driven byte-by-byte data transfer; supports broadcast mode and 10 bit addressing
- TPMx — One 6-channel and two 3-channel; Selectable input capture, output compare, or buffered edge- or center-aligned PWMs on each channel
- RTC — 8-bit modulus counter with binary or decimal based prescaler; External clock source for precise time base, time-of-day, calendar or task scheduling functions; Free running on-chip low power oscillator (1 kHz) for cyclic wake-up without external components
- Input/Output
  - 70 GPIOs and 1 input-only and 1 output-only pin
  - 16 KBI interrupts with selectable polarity
  - Hysteresis and configurable pull-up device on all input pins; Configurable slew rate and drive strength on all output pins.
  - SET/CLR registers on 16 pins (PTC and PTE)
  - 16 bits of Rapid GPIO connected to the CPU's high-speed local bus with set, clear, and toggle functionality

MCF51QE128 Reference Manual

Covers MCF51QE128
MCF51QE64
MCF51QE32

Related Documentation:

- **MCF51QE128 (Data Sheet)**
  Contains pin assignments and diagrams, all electrical specifications, and mechanical drawing outlines.

Find the most current versions of all documents at:

http://www.freescale.com

MCF51QE128RM
Rev. 3
09/2007

freescale™
semiconductor

Contents

| **Section Number** | **Title** | **Page** |

Chapter 1
Device Overview

Chapter 2
Pins and Connections

Chapter 3
Modes of Operation

MCF51QE128 MCU Series Reference Manual, Rev. 3

Get the latest version from freescale.com

Section Number	Title	Page

Chapter 4
Memory

Chapter 5
Resets, Interrupts, and General System Control

Chapter 6
Parallel Input/Output Control

Chapter 7
ColdFire Core

Chapter 8
Interrupt Controller (CF1_INTC)

Chapter 9
Rapid GPIO (RGPIO)

Chapter 10
Analog Comparator 3V (ACMPVLPV1)

Chapter 11
Analog-to-Digital Converter (S08ADC12V1)

MCF51QE128 MCU Series Reference Manual, Rev. 3

Get the latest version from freescale.com

Chapter 12
Internal Clock Source (S08ICSV3)

Chapter 13
Inter-Integrated Circuit (S08IICV2)

Chapter 14
Real-Time Counter (S08RTCV1)

Chapter 15
Serial Communications Interface (S08SCIV4)

Section Number	Title	Page

Chapter 16
Serial Peripheral Interface (S08SPIV3)

Chapter 17
Timer/Pulse-Width Modulator (S08TPMV3)

Chapter 18
Version 1 ColdFire Debug (CF1_DEBUG)

Appendix A
Revision History

Chapter 1
Device Overview

The MCF51QE128, MCF51QE64, and MCF51QE32 are members of the low-cost, low-power, high-performance Version 1 (V1) ColdFire family of 32-bit microcontroller units (MCUs). All MCUs in the family use the enhanced V1 ColdFire core and are available with a variety of modules, memory sizes, and package types. CPU clock rates on these devices can reach 50.33 MHz. Peripherals operate up to 25.165 MHz.

1.1   Devices in the MCF51QE128/64/32 Series

Table 1-1 summarizes the feature set available in the MCF51QE128/64/32 series of MCUs.

Table 1-1. MCF51QE128 Series Features by MCU and Package

Feature	MCF51QE128		MCF51QE64	MCF51QE32
Flash size (Kbytes)	128		64	32
RAM size (Kbytes)	8		8	8
Pin quantity	80	64	64	64
Version 1 ColdFire core with debug	yes			
ACMP1	yes			
ACMP2	yes			
ADC channels	24	20	20	20
ICS	yes			
IIC1	yes			
IIC2	yes			
KBI	16			
Port I/O[1, 2]	70	54	54	54
Rapid GPIO	yes			
COP	yes			
RTC	yes			
SCI1	yes			
SCI2	yes			
SPI1	yes			
SPI2	yes			
Interrupt Controller	yes			

MCF51QE128 MCU Series Reference Manual, Rev. 3

Table 1-1. MCF51QE128 Series Features by MCU and Package (continued)

Feature	MCF51QE128	MCF51QE64	MCF51QE32
External IRQ	yes		
Low-Voltage Detect (LVD)	yes		
TPM1 channels	3		
TPM2 channels	3		
TPM3 channels	6		
XOSC	yes		

[1] Port I/O count does not include the input-only PTA5/IRQ/TPM1CLK/$\overline{\text{RESET}}$ or the output-only PTA4/ACMP1O/BKGD/MS.

[2] 16 bits associated with Ports C and E are shadowed with ColdFire Rapid GPIO module.

1.2 MCU Block Diagram

The block diagram in Figure 1-1 shows the structure of the MCF51QE128/64/32 MCU.
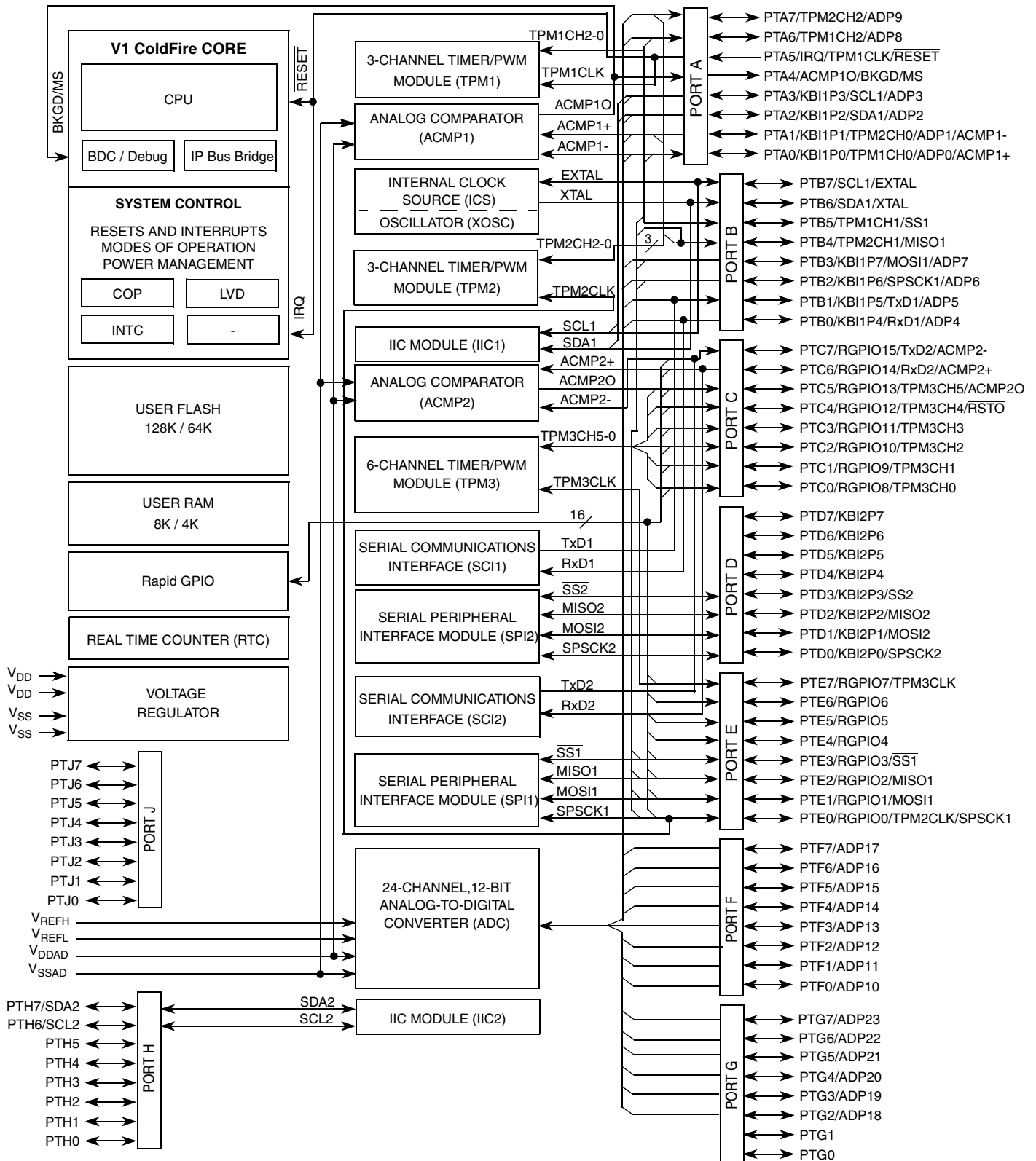
Figure 1-1. MCF51QE128/64/32 Block Diagram

Table 1-2 provides the functional version of the on-chip modules

Table 1-2. Module Versions

Module		Version
Analog Comparator	(ACMP)	4
Analog-to-Digital Converter	(ADC)	1
V1 ColdFire Core	(CF1_CORE)	1
V1 ColdFire Interrupt Controller	(CF1_INTC)	1
V1 ColdFire Debug Module	(CF1_DEBUG)	1
General Purpose I/O	(GPIO)	2
Inter-Integrated Circuit	(IIC)	2
Internal Clock Source	(ICS)	3
Keyboard Interrupt	(KBI)	2
Low-Power Oscillator	(OSCVLP)	1
Port Set/Clear	(PSC)	1
Rapid GPIO	(RGPIO)	1
Real-Time Counter	(RTC)	1
Serial Communications Interface	(SCI)	4
Serial Peripheral Interface	(SPI)	3
Timer Pulse Width Modulator	(TPM)	3
Voltage Regulator	(PMC*x*)	1

1.3   V1 ColdFire Core

The MCF51QE128/64/32 devices contain the Version 1 (V1) ColdFire core optimized for area and low-power. This CPU implements ColdFire instruction set architecture revision C (ISA_C):

- No hardware support for MAC/EMAC and DIV instructions[1]
- Provides upward compatibility to all other ColdFire cores (V2–V5)

For more details on the V1 ColdFire core, see Chapter 7, "ColdFire Core".

1.4   System Clocks

This section discusses on-chip clock generation and distribution for the MCF51QE128/64/32 devices.

1.4.1   Internal Clock Source (ICS) Module

Figure 1-2 shows a simplified view of the internal clock source module. For clarity, only one of three available FLL modules is shown.

---

1. These operations can be emulated via software functions.
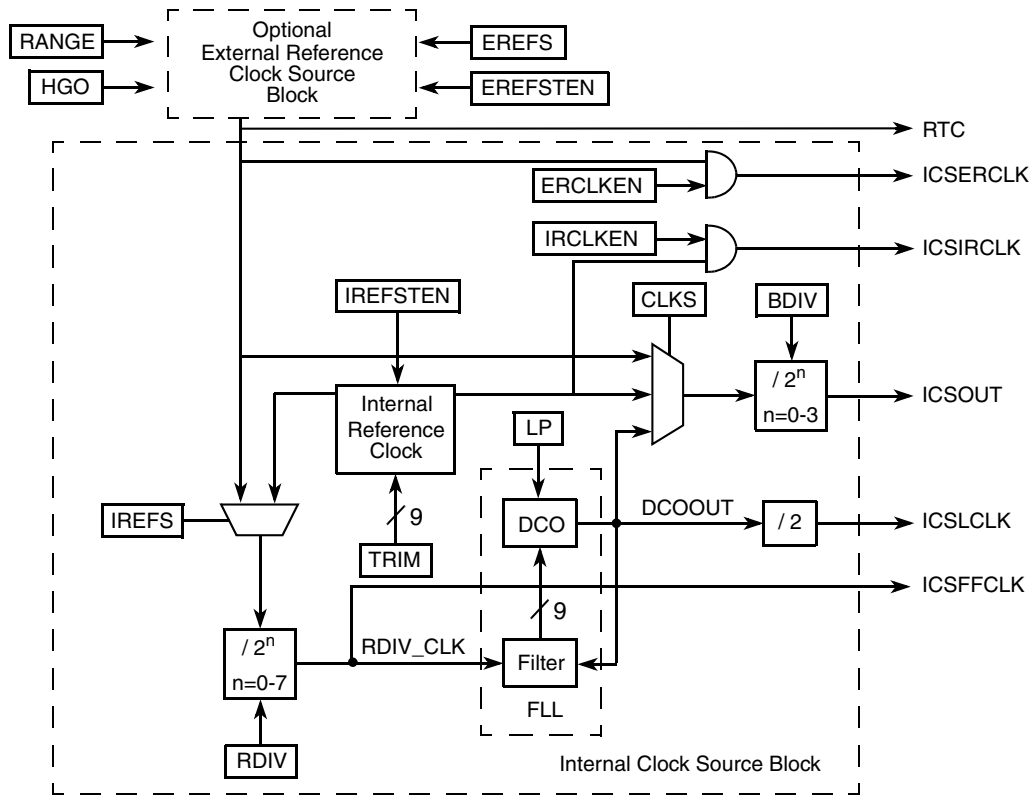
Figure 1-2. Simplified ICS Block Diagram

1.4.2 System Clock Distribution

Figure 1-3 shows a simplified clock connection diagram. Some modules in the MCU have selectable clock inputs as shown. The clock inputs to the modules indicate the clock(s) used to drive the module function. All memory mapped registers associated with the modules (except RGPIO) are clocked with the peripheral clock (BUSCLK). The RGPIO registers are clocked with the CPU clock (ICSOUT). With the exception of the oscillator clock supplied directly to the RTC, the ICS supplies all clock sources:

- ICSOUT — This clock source is used throughout the core including the CPU. For consistency, it is known simply as the CPU clock. It is divided by two to generate the peripheral bus clock. Control bits in the ICS control registers determine which of three clock sources is connected:
  - Internal reference clock
  - External reference clock
  - Frequency-locked loop (FLL) output

  See Chapter 12, "Internal Clock Source (S08ICSV3)," for details on configuring the ICSOUT clock.

- ICSLCLK — This clock source is derived from the 10/20 MHz DCO (digitally controlled oscillator) of the ICS when the ICS is configured to run off of the internal or external reference clock. Development tools can select this internal self-clocked source (~10 MHz) to speed up BDC communications in systems where the bus clock is slow.
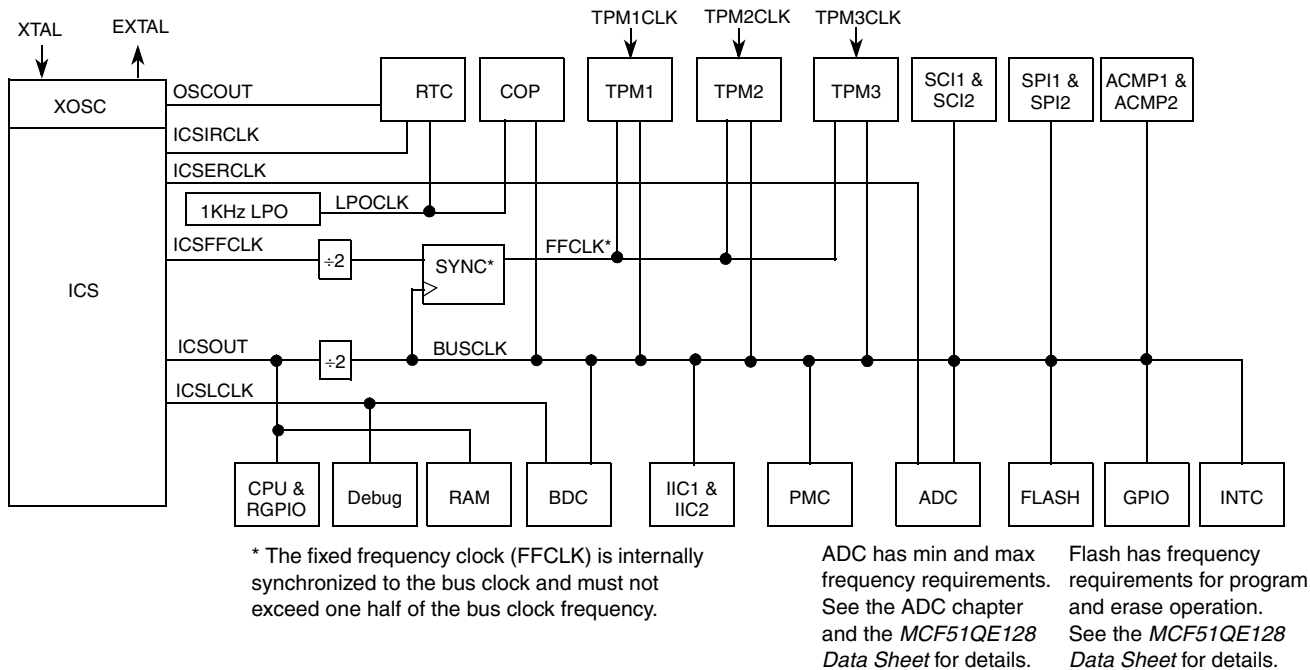
Figure 1-3. System Clock Distribution Diagram

- OSCOUT — This is the direct output of the external oscillator module and can be selected as the real-time counter clock source. See Chapter 14, "Real-Time Counter (S08RTCV1)," for more information.

- ICSERCLK — This is the external reference clock and can be selected as the real-time counter clock source or the alternate clock for the ADC module. Section 11.4.7, "External Reference Clock," explains the ICSERCLK in more detail. See Chapter 11, "Analog-to-Digital Converter (S08ADC12V1)," for more information regarding the use of ICSERCLK with this module.

- ICSIRCLK — This is the internal reference clock and can be selected as the real-time counter clock source. Section 11.4.6, "Internal Reference Clock," explains the ICSIRCLK in more detail. See Chapter 14, "Real-Time Counter (S08RTCV1)," for more information regarding the use of ICSIRCLK.

- ICSFFCLK — This generates the fixed frequency clock (FFCLK) after being synchronized to the bus clock. It can be selected as clock source for the TPM modules. The frequency of the ICSFFCLK is determined by the settings of the ICS. See Section 11.4.8, "Fixed Frequency Clock," for details.

- LPOCLK — This clock is generated from an internal low-power oscillator that is completely independent of the ICS module. The LPOCLK can be selected as the clock source to the RTC or COP modules. See Chapter 14, "Real-Time Counter (S08RTCV1)," and Section 5.3.1, "Computer Operating Properly (COP) Watchdog," for details on using the LPOCLK with these modules.

- TPM*x*CLK — TPM*x*CLKs are optional external clock sources for the TPM modules. The TPM*x*CLK must be limited to 1/4 the frequency of the bus clock for synchronization. See Section 16.2.1, "External TPM Clock Sources," for more details.

The ADC module also has an internally generated asynchronous clock that allows it to run in stop mode (ADACK). This signal is not available externally and is not shown in this figure.

1.4.3 ICS Modes of Operation

There are seven modes of operation for the internal clock source (ICS) module: FEI, FEE, FBI, FBILP, FBE, FBELP, and stop. These are shown in Figure 1-4. The IREFS and CLKS fields are contained within the ICS module definition. The LP bit is part of the on-chip power management controller (PMC) block. It is the responsibility of the software to ensure that the system bus frequency is less than 125 kHz and the FLLs are disengaged prior to enabling switching the LP bit to enable FBELP and FBILP modes of operation.

The clock source for the BDC is controlled by the debug CLKSW bit, discussed later in this document. Choices for the BDC clock are ICSOUT and the output from the 10MHz bus / 20 MHz CPU clock FLL.

1.4.3.1 FLL Engaged Internal (FEI)

In FLL engaged internal mode, which is the default mode, the ICS supplies a clock derived from one of three on chip FLLs, which are controlled by the internal reference clock. Upon exiting reset, the default FLL generates the 10 MHz bus/20 MHz CPU clocks.

1.4.3.2 FLL Engaged External (FEE)

In FLL engaged external mode, the ICS supplies a clock derived from one of the three FLLs, which are controlled by an external reference clock.

1.4.3.3 FLL Bypassed Internal (FBI)

In FLL bypassed internal mode, the FLLs are enabled and controlled by the internal reference clock, but are bypassed. The ICS supplies a clock derived from the internal reference clock.

1.4.3.4 FLL Bypassed Internal Low-Power (FBILP)

In FLL bypassed internal low-power mode, the FLLs are disabled and bypassed, and the ICS supplies a clock derived from the internal reference clock.

1.4.3.5 FLL Bypassed External (FBE)

In FLL bypassed external mode, the FLLs are enabled and controlled by an external reference clock, but are bypassed. The ICS supplies a clock derived from the external reference clock. The external reference clock can be an external crystal/resonator supplied by an OSC controlled by the ICS, or it can be another external clock source.

IREFS=1
CLKS=00
LP=0

IREFS=0
CLKS=10
LP=0

FLL Engaged
Internal (FEI)

IREFS=1
CLKS=01
LP=0

FLL Bypassed
External Low-
Power(FBELP)

FLL Bypassed
External (FBE)

FLL Bypassed
Internal (FBI)

FLL Bypassed
Internal Low-
Power(FBILP)

IREFS=0
CLKS=10
LP=1

IREFS=1
CLKS=01
LP=1

FLL Engaged
External (FEE)

IREFS=0
CLKS=00
LP=0

Entered from any state when
MCU enters stop with
ENBDM=0.

Stop

Returns to state that was active
before MCU entered stop, unless
reset occurs while in stop.
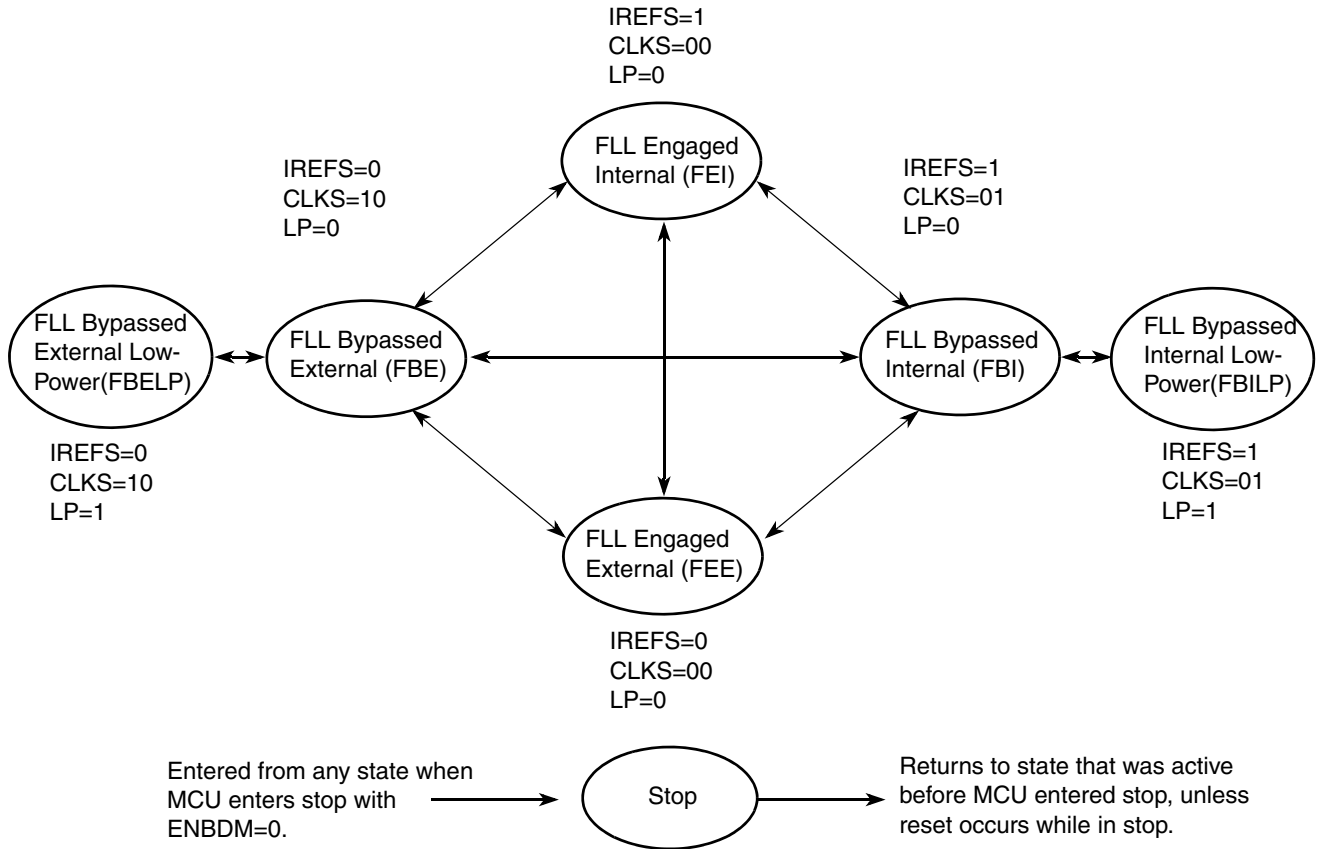
Figure 1-4. ICS Modes of Operation

1.4.3.6   FLL Bypassed External Low-Power (FBELP)

In FLL bypassed external low-power mode, the FLLs are disabled and bypassed, and the ICS supplies a
clock derived from the external reference clock. The external reference clock can be an external
crystal/resonator supplied by an OSC controlled by the ICS, or it can be another external clock source.

1.4.3.7   Stop (STOP)

In stop mode, the FLLs are disabled and the internal or external reference clocks can be selected to be
enabled or disabled. The ICS does not provide an MCU clock source unless the debug ENBDM bit is set.

Chapter 2
Pins and Connections

This section describes signals that connect to package pins. It includes pinout diagrams, recommended system connections, and detailed discussions of signals.

2.1   Device Pin Assignment

Figure 2-1 shows the 80-pin assignments for the MCF51QE128 devices. Figure 2-2 shows the 64-pin assignments for the MCF51QE128/64/32 devices.

Pins in **bold** are added from the next smaller package.

Figure 2-1. 80-Pin LQFP
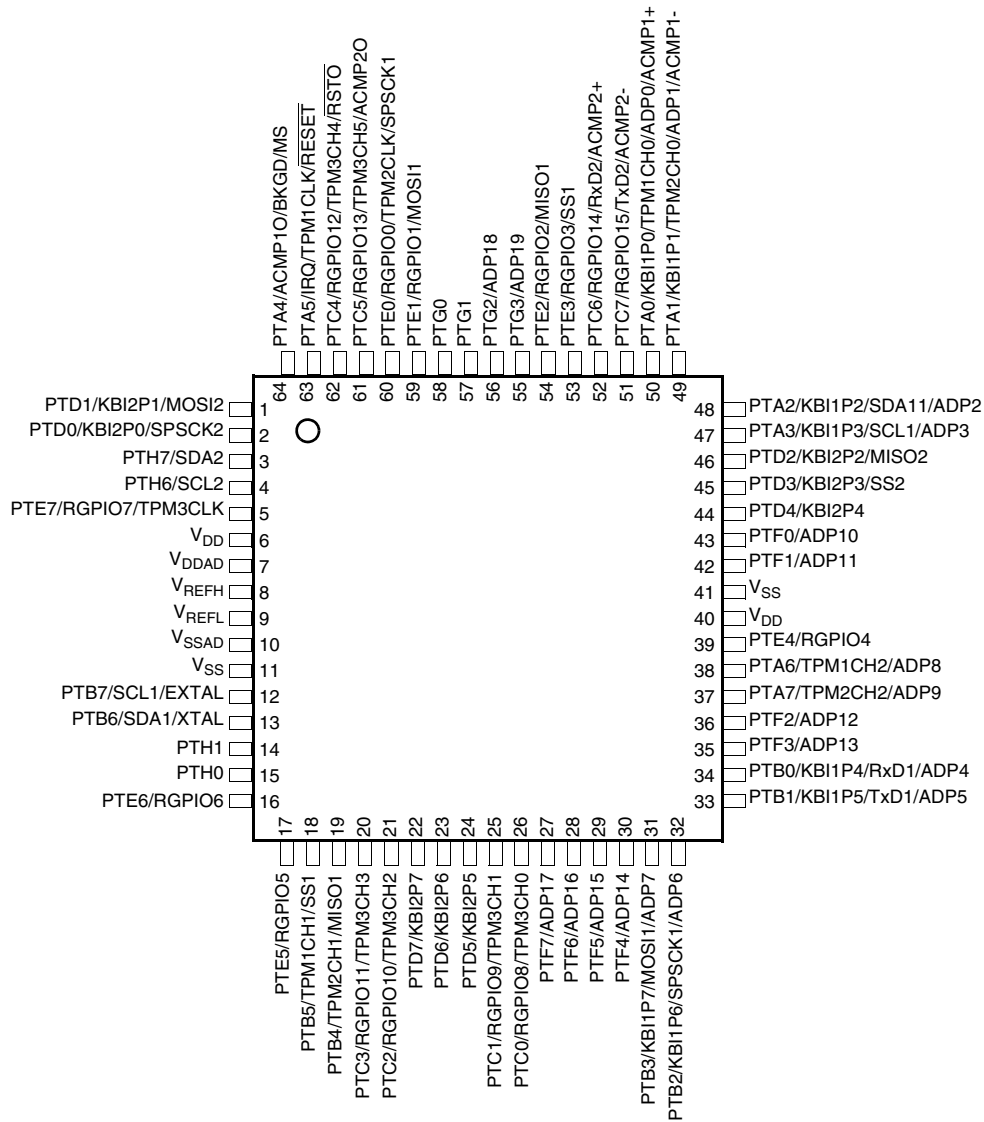
Figure 2-2. 64-Pin LQFP

2.2 Recommended System Connections

Figure 2-3 shows pin connections common to MCF51QE128/64/32 application systems.
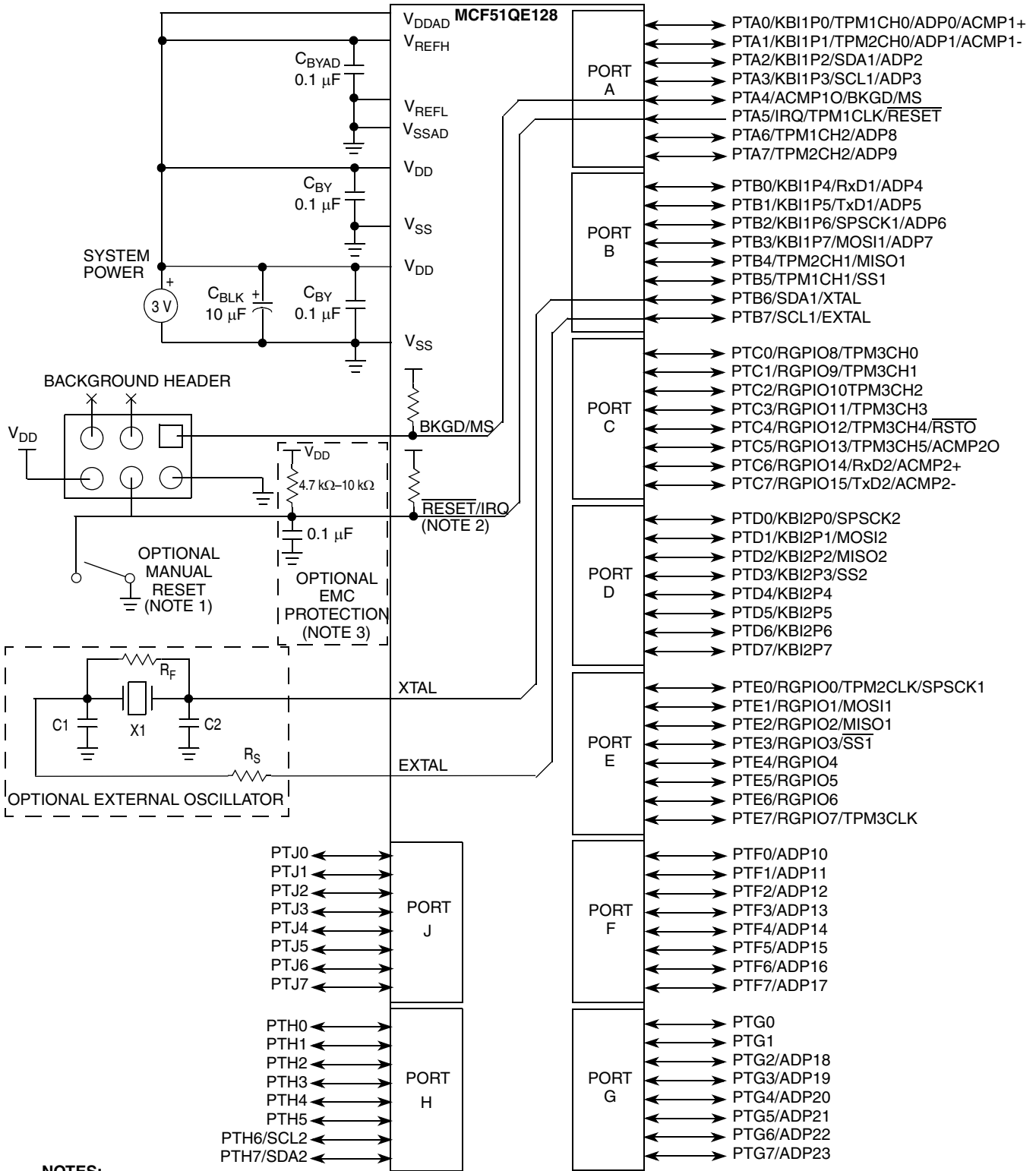
NOTES:
1 $\overline{\text{RESET}}$ pin can only be used to reset into user mode; it cannot be used to enter BDM. Entry into BDM is accomplished by holding BGKD low during POR or setting CSR2[BDFR] with BGKD held low after issuing the BDM write command.
2 $\overline{\text{RESET}}$/IRQ features have optional internal pullup device.
3 RC filter on $\overline{\text{RESET}}$/IRQ pin recommended for noisy environments.

Figure 2-3. Basic System Connections

MCF51QE128 MCU Series Reference Manual, Rev. 3

2.2.1 Power

V_{DD} and V_{SS} are the primary power supply pins for the MCU. This voltage source supplies power to all I/O buffer circuitry and to an internal voltage regulator. The internal voltage regulator provides regulated lower-voltage source to the CPU and other internal circuitry of the MCU.

Typically, application systems have two separate capacitors across the power pins. In this case, there should be a bulk electrolytic capacitor, such as a 10µF tantalum capacitor, to provide bulk charge storage for the overall system and a 0.1µF ceramic bypass capacitor located as close to the MCU power pins as practical to suppress high-frequency noise. The MCF51QE128/64/32has two V_{DD} pins. Each pin must have a bypass capacitor for best noise suppression.

V_{DDAD} and V_{SSAD} are the analog power supply pins for the MCU. This voltage source supplies power to the ADC module. A 0.1µF ceramic bypass capacitor should be located as close to the MCU power pins as practical to suppress high-frequency noise.

2.2.2 Oscillator

Immediately after reset, the MCU uses an internally generated clock provided by the internal clock source (ICS) module. For more information on the ICS, see Chapter 12, "Internal Clock Source (S08ICSV3)".

The oscillator (XOSC) in this MCU is a Pierce oscillator that can accommodate a crystal or ceramic resonator. Optionally, an external clock source can be connected to the EXTAL input pin.

Refer to Figure 2-3 for the following discussion. R_S (when used) and R_F should be low-inductance resistors such as carbon composition resistors. Wire-wound resistors, and some metal film resistors, have too much inductance. C1 and C2 normally should be high-quality ceramic capacitors specifically designed for high-frequency applications.

R_F is used to provide a bias path to keep the EXTAL input in its linear range during crystal startup; its value is not generally critical. Typical systems use 1 MΩ to 10 MΩ. Higher values are sensitive to humidity and lower values reduce gain and (in extreme cases) could prevent startup.

C1 and C2 are typically in the 5pF to 25pF range and are chosen to match the requirements of a specific crystal or resonator. Be sure to take into account printed circuit board (PCB) capacitance and MCU pin capacitance when selecting C1 and C2. The crystal manufacturer typically specifies a load capacitance that is the series combination of C1 and C2 (which are usually the same size). As a first-order approximation, use 10pF as an estimate of combined pin and PCB capacitance for each oscillator pin (EXTAL and XTAL).

2.2.3 $\overline{\text{RESET}}$ and $\overline{\text{RSTO}}$

After a power-on reset (POR), the PTA5/IRQ/TPM1CLK/$\overline{\text{RESET}}$ pin defaults to a general-purpose input port pin, PTA5. Setting RSTPE in SOPT1 configures the pin to be the $\overline{\text{RESET}}$ pin. After configured as $\overline{\text{RESET}}$, the pin remains $\overline{\text{RESET}}$ until the next POR. The $\overline{\text{RESET}}$ pin can be used to reset the MCU from an external source when the pin is driven low. When enabled as the $\overline{\text{RESET}}$ pin (RSTPE = 1), the pin is configured as an input with an internal pullup device automatically enabled.

NOTE

The $\overline{RESET}$ pin does not contain a clamp diode to V_{DD} and should not be driven above V_{DD}.

NOTE

In EMC-sensitive applications, an external RC filter is recommended on the $\overline{RESET}$ pin, if enabled. See Figure 2-3 for an example.

After a power-on reset (POR), the PTC4/RGPIO12/TPM3CH4/$\overline{RSTO}$ pin defaults to a general-purpose port pin, PTC4. Setting RSTOPE in SOPT1 configures the pin as an open drain with internal pullup acting as reset out ($\overline{RSTO}$). The $\overline{RSTO}$ pin reflects the current state of the internal MCU reset signal. As long as the MCU is not in a reset state, the $\overline{RSTO}$ pin is driven high. When an internal reset occurs and RSTPE is set, the $\overline{RSTO}$ pin is pulled low for as long as the internal reset signal is low. This allows other devices in the system to detect the MCU's reset state.

When enabled as the $\overline{RSTO}$ pin (RSTOPE = 1), the pin is automatically configured as an output only. The $\overline{RSTO}$ pin can be enabled independently of the $\overline{RESET}$ pin. After being configured as $\overline{RSTO}$, the pin remains in this mode until the next POR.

NOTE

The $\overline{RSTO}$ pullup should not be used as a pullup for components external to the MCU. Inputs to internal gates connected to this pin are resistively pulled high, but VDD is not seen at the pin itself.

2.2.4   Background / Mode Select (BKGD/MS)

During a power-on-reset (POR) or background debug force reset (see the BDFR bit in Section 18.3.3, "Configuration/Status Register 2 (CSR2)," for more information), the PTA4/ACMP1O/BKGD/MS pin functions as a mode select pin. Immediately after any reset, the pin functions as the background data pin and can be used for background debug communication.

The debug communication function is enabled when SOPT1[BKGDPE] is set. When enabled as the BKGD/MS pin (BKGDPE = 1), an internal pullup device is automatically enabled. BKGDPE is set following any reset of the MCU and must be cleared to use the PTA4/ACMP1O/BKGD/MS pin's alternative pin functions.

If this pin is unconnected, the MCU enters normal operating mode at the rising edge of the internal reset after a POR or forced BDC reset. If a debug system is connected to the 6-pin standard background debug header, it can hold BKGD/MS low during a POR or immediately after issuing a background debug force reset[1], which forces the MCU to halt mode.

The BKGD/MS pin is used primarily for background debug controller (BDC) communications using a custom protocol that uses 16 clock cycles of the target MCU's BDC clock per bit time. The target MCU's BDC clock can be as fast as the bus clock rate, so there should never be any significant capacitance connected to the BKGD/MS pin that could interfere with background serial communications.

---

1. Specifically, BKGD must be held low through the first 16 cycles after deassertion of the internal reset.

MCF51QE128 MCU Series Reference Manual, Rev. 3

Although the BKGD/MS pin is a pseudo open-drain pin, the background debug communication protocol provides brief, actively driven, high speed-up pulses to ensure fast rise times. Small capacitances from cables and the absolute value of the internal pullup device play almost no role in determining rise and fall times on the BKGD/MS pin.

2.2.5 ADC Reference Pins (V$_{REFH}$, V$_{REFL}$)

The V$_{REFH}$ and V$_{REFL}$ pins are the voltage reference high and voltage reference low inputs, respectively, for the ADC module.

2.2.6 General-Purpose I/O and Peripheral Ports

The MCF51QE128/64/32series of MCUs support up to 70 general-purpose I/O pins, 1 input-only pin, and 1 output-only pin, which are shared with on-chip peripheral functions (timers, serial I/O, ADC, ACMP, etc.).

When a port pin is configured as a general-purpose output or a peripheral uses the port pin as an output, software can select one of two drive strengths and enable or disable slew rate control. When a port pin is configured as a general-purpose input or a peripheral uses the port pin as an input, software can enable a pull-up device. Immediately after reset, all of these pins are configured as high-impedance general-purpose inputs with internal pull-up devices enabled.

PTC4 is a special case I/O pin. When the PTC4/RGPIO12/TPM3CH4/$\overline{\text{RSTO}}$ pin is configured as $\overline{\text{RSTO}}$, it is an open drain output with an internal pullup. The voltage observed on the pin is not pulled to VDD, and an external pullup resistor is recommended if this pin must drive off-chip signals.

PTA5/IRQ/TPM1CLK/$\overline{\text{RESET}}$ is also a special case I/O pin. It can only be configured as an input.

When an on-chip peripheral system is controlling a pin, data direction control bits still determine what is read from the port data registers, even though the peripheral controls the pin direction via the pin's output buffer enable. For information about controlling these pins as general-purpose I/O pins, see Chapter 6, "Parallel Input/Output Control".

NOTE

To avoid extra current drain from floating input pins, the reset initialization routine in the application program should enable on-chip pullup devices or change the direction of unused or non-bonded pins to outputs so they do not float.

Table 2-1. Pin Assignment by Package and Pin Sharing Priority

Pin Number		Lowest	←——	**Priority**	——→	Highest
80	**64**	**Port Pin**	**Alt 1**	**Alt 2**	**Alt 3**	**Alt 4**
1	1	PTD1	KBI2P1	MOSI2		
2	2	PTD0	KBI2P0	SPSCK2		
3	3	PTH7	SDA2			
4	4	PTH6	SCL2			
5	—	PTH5				
6	—	PTH4				
7	5	PTE7	RGPIO7	TPM3CLK		
8	6					V_{DD}
9	7					V_{DDAD}
10	8					V_{REFH}
11	9					V_{REFL}
12	10					V_{SSAD}
13	11					V_{SS}
14	12	PTB7	SCL1			EXTAL
15	13	PTB6	SDA1			XTAL
16	—	PTH3				
17	—	PTH2				
18	14	PTH1				
19	15	PTH0				
20	16	PTE6	RGPIO6			
21	17	PTE5	RGPIO5			
22	18	PTB5	TPM1CH1	SS1		
23	19	PTB4	TPM2CH1	MISO1		
24	20	PTC3	RGPIO11	TPM3CH3		
25	21	PTC2	RGPIO10	TPM3CH2		
26	22	PTD7	KBI2P7			
27	23	PTD6	KBI2P6			
28	24	PTD5	KBI2P5			
29	—	PTJ7				
30	—	PTJ6				
31	—	PTJ5				
32	—	PTJ4				
33	25	PTC1	RGPIO9	TPM3CH1		
34	26	PTC0	RGPIO8	TPM3CH0		
35	27	PTF7				ADP17
36	28	PTF6				ADP16
37	29	PTF5				ADP15
38	30	PTF4				ADP14
39	31	PTB3	KBI1P7	MOSI1[1]		ADP7
40	32	PTB2	KBI1P6	SPSCK1		ADP6
41	33	PTB1	KBI1P5	TxD1		ADP5
42	34	PTB0	KBI1P4	RxD1		ADP4
43	—	PTJ3				

Table 2-1. Pin Assignment by Package and Pin Sharing Priority (continued)

Pin Number		Lowest	⟵	**Priority**	⟶	Highest
80	**64**	**Port Pin**	**Alt 1**	**Alt 2**	**Alt 3**	**Alt 4**
44	—	PTJ2				
45	35	PTF3				ADP13
46	36	PTF2				ADP12
47	37	PTA7	TPM2CH2			ADP9
48	38	PTA6	TPM1CH2			ADP8
49	39	PTE4	RGPIO4			
50	40					VDD
51	41					VSS
52	42	PTF1				ADP11
53	43	PTF0				ADP10
54	—	PTJ1				
55	—	PTJ0				
56	44	PTD4	KBI2P4			
57	45	PTD3	KBI2P3	SS2		
58	46	PTD2	KBI2P2	MISO2		
59	47	PTA3	KBI1P3	SCL1[2]		ADP3
60	48	PTA2	KBI1P2	SDA1		ADP2
61	49	PTA1	KBI1P1	TPM2CH0	ADP1	ACMP1-
62	50	PTA0	KBI1P0	TPM1CH0	ADP0	ACMP1+
63	51	PTC7	RGPIO15	TxD2		ACMP2-
64	52	PTC6	RGPIO14	RxD2		ACMP2+
65	—	PTG7				ADP23
66	—	PTG6				ADP22
67	—	PTG5				ADP21
68	—	PTG4				ADP20
69	53	PTE3	RGPIO3	SS1		
70	54	PTE2	RGPIO2	MISO1		
71	55	PTG3				ADP19
72	56	PTG2				ADP18
73	57	PTG1				
74	58	PTG0				
75	59	PTE1	RGPIO1	MOSI1		
76	60	PTE0	RGPIO0	TPM2CLK	SPSCK1	
77	61	PTC5	RGPIO13	TPM3CH5		ACMP2O
78	62	PTC4	RGPIO12	TPM3CH4	RSTO	
79	63	PTA5	IRQ	TPM1CLK	RESET	
80	64	PTA4[3]	ACMP1O	BKGD	MS	

[1] SPI1 pins (SS1, MISO1, MOSI1, and SPSCK2) can be repositioned using SOPT2[SPI1PS]. Default locations are PTB5, PTB4, PTB3, and PTB2, respectively.

MCF51QE128 MCU Series Reference Manual, Rev. 3

[2] IIC1 pins (SCL1 and SDA1) can be repositioned using SOPT2[IIC1PS]. Default locations are PTA3 and PTA2, respectively.

[3] PTA4/ACMP1O/BKGD/MS is limited to output-only for the port I/O function.

Chapter 3
Modes of Operation

3.1 Introduction

The operating modes of the MCF51QE128/64/32 are described in this chapter. Entry into each mode, exit from each mode, and functionality while in each of the modes are described.

The overall system mode is generally a function of a number of separate, but inter-related variables: debug mode, security mode, power mode, and clock mode. Clock modes were discussed in Section 1.4.3, "ICS Modes of Operation". This chapter explores the other dimensions of the system operating mode.

3.2 Features

- Debug mode for code development. For V1 ColdFire devices, such as MCF51QE128/64/32 , debug mode is mutually exclusive with use of secure mode (next item).
- Secure mode — BDC access to CPU resources is extremely restricted. It is possible to tell that the device has been secured, and to clear security, which involves mass erasing the on-chip flash memory. No other CPU access is allowed. Secure mode can be used in conjunction with each of the power modes below.
- Run mode — CPU clocks can be run at full speed and the internal supply is fully regulated.
- LPrun mode — CPU and peripheral clocks are restricted to 250 kHz CPU clock and 125 kHz bus clock maximum and the internal supply is in soft regulation.
- Wait mode — CPU shuts down to conserve power; peripheral clocks are running and full regulation is maintained.
- LPwait mode — CPU shuts down to conserve power; peripheral clocks are running at reduced speed (125 kHz maximum) and the internal voltage regulator is running in loose regulation mode.
- Stop modes — System (CPU and peripheral) clocks are stopped.
  - Stop4 — All internal circuits are powered (full regulation mode) and internal clock sources still at max frequency for fastest recovery.
  - Stop3 — All internal circuits are loosely regulated and clocks sources are at minimal values (125 kHz maximum), providing a good compromise between power utilization and speed of recovery.
  - Stop2 — Partial power-down of internal circuits; RAM content is retained. The lowest power mode for this device. A reset is required to return from stop2 mode.

On the MCF51QE128/64/32 , wait, stop2, stop3, and stop4 are all entered via the CPU STOP instruction. See Table 3-1, Figure 3-2, and subsequent sections of this chapter for details.

3.3   Overview

The ColdFire CPU has two primary user modes of operation, run and stop. (The CPU also supports a halt mode that is used strictly for debug operations.) The STOP instruction is used to invoke stop and wait modes for this family of devices.

If the WAITE control bit is set when STOP is executed, the wait mode is entered. Otherwise, if the STOPE bit is set, the CPU enters one of the stop modes. It is illegal to execute a STOP instruction if neither STOPE or WAITE are set. This results in reset assertion if CPUCR[IRD] is cleared or an illegal instruction exception if CPUCR[IRD] is set.

The MCF51QE128/64/32 devices augment stop, wait, and run in a number of ways. The power management controller (PMC) can run the device in fully-regulated mode, standby mode, and partial power-down mode. Standby (loose regulation) or partial power-down can be programmed to occur naturally as a result of a STOP instruction. Additionally, standby mode can be explicitly invoked via the LPR (low-power) bit in the PMC. Use of standby is limited to bus frequencies less than 125 kHz; and neither standby nor partial power-down are allowed when the ENBDM bit is set to enable debugging in stop and wait modes.

During partial power-down mode, the regulator is in standby mode and much of the digital logic on the chip is switched off. These interactions can be seen schematically in Figure 3-1. This figure is for conceptual purposes only. It does not reflect any sequence or time dependencies between the PMC and other parts of the device, nor does it represent any actual design partitioning.
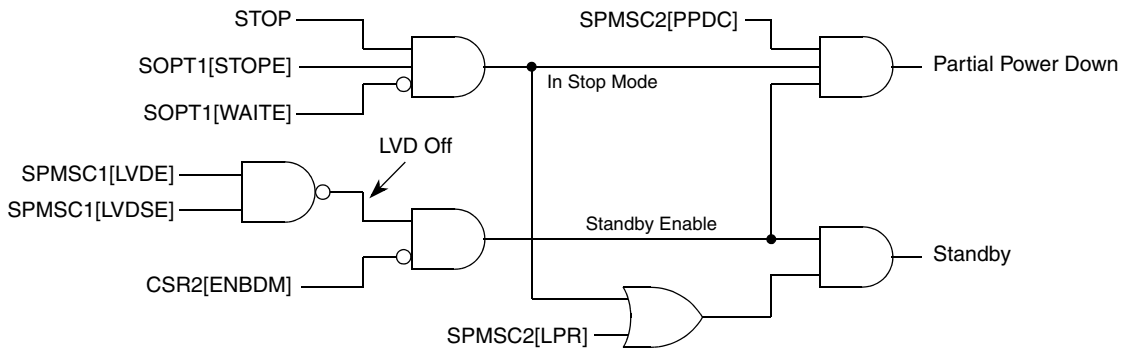
Figure 3-1. MCF51QE128/64/32 Power Modes - Conceptual Drawing

It is illegal for the software to have PPDC and LPR asserted concurrently. This restriction arises because the sequence of events from normal to low-power modes involves use of both bits. After entering a low-power mode, it is not possible to switch to another low-power mode.

Table 3-1. CPU / Power Mode Selections

Mode of Operation	SOPT1 SIM STOPE	SOPT1 SIM WAITE	CSR2 BDC ENBDM[1]	SPMSC1 PMC LVDE	SPMSC1 PMC LVDSE	SPMSC2 PMC LPR	SPMSC2 PMC PPDC	CPU and Peripheral Clocks	BDC Clock	Switched Power
Run mode - processor and peripherals clocked normally.			x	x	x	0	x	On. ICS in any mode	On **Note:** When not needed, the BDC clock can be gated off at the discretion of the processor. The clock is available within a few cycles of demand by the processor, normally when a negative edge is detected on BKGD. The BDM command associated with that negative edge may not take affect.	On
	x	x	x	1	1	x	x			
			1	x	x	x	x			
LPrun mode with low voltage detect disabled - processor and peripherals clocked at low frequency[2]. Low voltage detects are not active.	x	x	0	0	x	1	0	Low freq required. ICS in FBELP mode.		Loose Reg
			0	1	0					
Wait mode - processor clock nominally inactive, but peripherals are clocked.	x	1	x	x	x	0	x	Periph clocks on. CPU clock on if ENBDM=1.		On
			x	1	1	x	x			
			1	x	x	x	x	On		
LPwait mode - processor clock is inactive, peripherals are clocked at low frequency and the PMC is loosely regulating. Low voltage detects are not active.	x	1	0	0	x	1	0	CPU clock is off. Periph clocks at low speed. ICS in FBELP.		Loose Reg
				1	0					
Stop modes disabled; Illegal opcode reset if STOP instruction executed and CPUCR[IRD] is cleared, else illegal instruction exception is generated.	0	0	Function of BKGD/MS at reset	⇒1	⇒1	⇒0	⇒0	⇒On	Function of BKGD/MS at reset	⇒On
Stop4 - Either low-power modes have not been requested, or low voltage detects are enabled or ENBDM = 1.	1	0	x	x	x	0	0	Peripheral clocks off. CPU clock on if ENBDM=1.	BDC clock enabled only if ENBDM=1 prior to entering stop.	On
			x	1	1	1	0			
			x	1	1	0	1			
			1	x	x	x	x	CPU clock on. Periph clocks off.		
Stop3 - Low voltage detect in stop is not enabled. Clocks must be at low frequency and are gated. The regulator is in loose regulation.	1	0	0	1	0	1	0	Low freq required. ICS in FBELP mode. CPU and peripheral clocks are gated off.	Off	Loose Reg
				0	x					
Stop2 - Low voltage detects are not active. If BDC is enabled, stop4 is invoked rather than stop2.	1	0		1	0	0	1	N/A	N/A	Off
				0	x					

[1] ENBDM is located in the upper byte of the XCSR register which is write-accessible only through BDC commands, see Section 18.3.2, "Extended Configuration/Status Register (XCSR)".

[2] 250 kHz maximum CPU frequency in LPrun; 125 kHz maximum peripheral clock frequency.

Mode	Regulator State
Run	Full On
Wait	Full On
Stop4	Full On
LPrun	Standby
LPwait	Standby
Stop3	Standby
Stop2	Partial Power Off
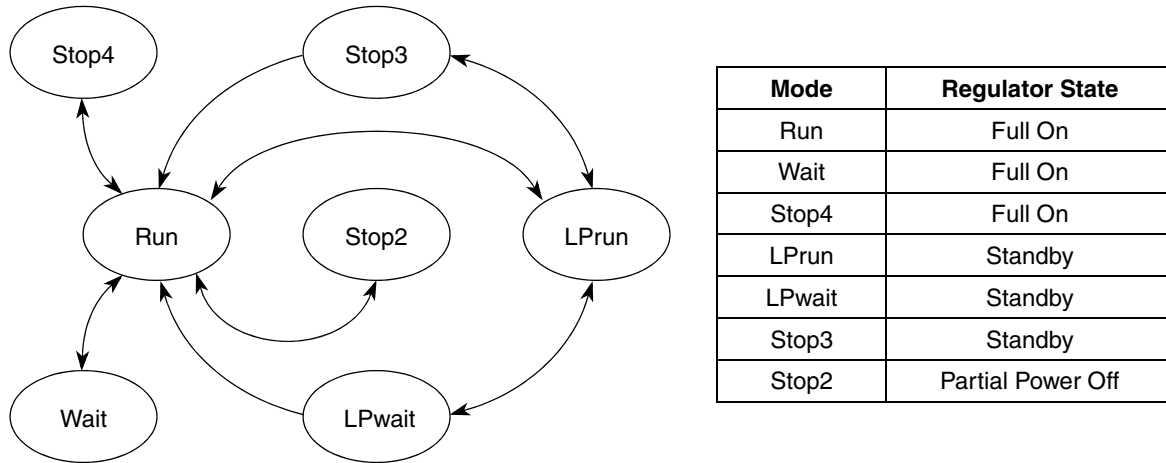
Figure 3-2. Allowable Power Mode Transitions for Mission Mode MCF51QE128/64/32

Figure 3-2 illustrates mission mode state transitions allowed between the legal states shown in Table 3-1. PTA5/IRQ/TPM1CLK/$\overline{\text{RESET}}$ must be asserted low to exit stop2. Only interrupt assertion is necessary to exit the other stop and wait modes.

Figure 3-3 takes the same set of states and transitions shown in Figure 3-2 and adds the BDM halt mode for development purposes. If BDM is enabled, the chip automatically shifts LP modes into their fully regulated equivalents. If software or debugger set the LPR bit in SPMSC2 while BDM is enabled, the LPRS bit reflects the fact that the regulator is not in standby. Similarly, the PPDF does not indicate a recovery from stop2 if ENBDM forced stop4 to occur in its place.[1]

Stated another way, if ENBDM has been set via the BDM interface, then the power management controller keeps (or puts) the regulator in full regulation despite other settings in the contrary. The states shown in Figure 3-3 then map as follows:

- LPrun $\Rightarrow$ Run
- LPwait $\Rightarrow$ Wait
- Stop3 $\Rightarrow$ Stop4
- Stop2 $\Rightarrow$ Stop4

From a software perspective (and disregarding PMC status bits), the system remains in the appropriate low-power state, and can be debugged as such.

See Section 3.7, "Wait Modes," for a description of the various ways to enter halt mode.

---

1. This can have subtle impacts on recovery from stop. The IRQ input can wake the device from stop4 if it has been enabled for that purpose. That same pin wakes the device from stop2 even when the IRQ is not enabled (there is an asynchronous path to the power management controller in that state).
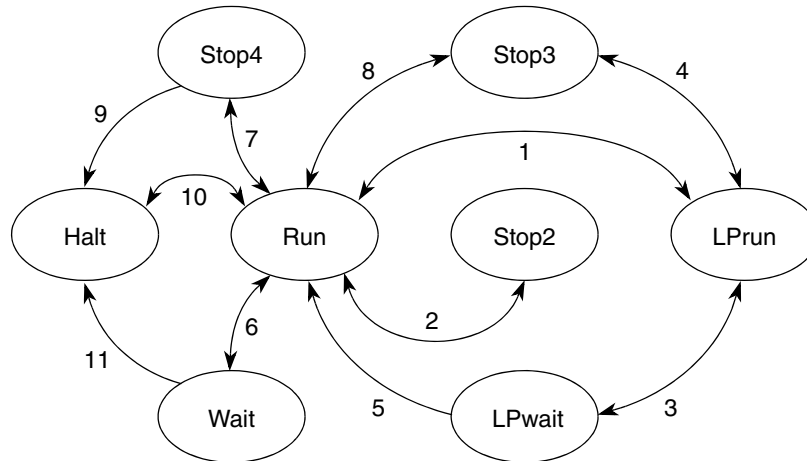
Figure 3-3. All Allowable Power Mode Transitions for MCF51QE128/64/32

Table 3-2 defines triggers for the various state transitions shown in Figure 3-2.

Table 3-2. Triggers for Transitions Shown in Figure 3-2

Transition #	From	To	Trigger
1	Run	LPrun	Configure settings shown in Table 3-1, switch LPR=1 last
	LPrun	Run	Clear LPR
			Interrupt when LPWUI=1
			Negative transition on enabled BKGD/MS pin.
2	Run	Stop2	Pre-configure settings shown in Table 3-1, execute STOP instruction
	Stop2	Run	Assert zero on PTA5/IRQ/TPM1CLK/$\overline{\text{RESET}}$[1] or RTC timeout. Reload environment from RAM
3	LPrun	LPwait	Pre-configure settings shown in Table 3-1, execute STOP instruction
	LPwait	LPrun	Interrupt when LPWUI=0
4	LPrun	Stop3	Execute STOP instruction
	Stop3	LPrun	Interrupt when LPWUI=0
5	LPwait	Run	Interrupt when LPWUI=1
	Run	LPwait	Not supported.
6	Run	Wait	Pre-configure settings shown in Table 3-1, execute STOP instruction
	Wait	Run	Interrupt
7	Run	Stop4	Pre-configure settings shown in Table 3-1, execute STOP instruction
	Stop4	Run	Interrupt

Table 3-2. Triggers for Transitions Shown in Figure 3-2 (continued)

Transition #	From	To	Trigger
8	Stop3	Run	Interrupt when LPWUI=1
	Run	Stop3	Pre-configure settings shown in Table 3-1, execute STOP instruction
9	Stop4	Halt	When a BACKGROUND command is received through the BKGD/MS pin (ENBDM must equal one).
	Halt	Stop4	Not supported.
10	Halt	Run	GO instruction issued via BDM
	Run	Halt	When a BACKGROUND command is received through the BKGD/MS pin OR When a HALT instruction is executed OR When encountering a BDM breakpoint
11	Wait	Halt	When a BACKGROUND command is received through the BKGD/MS pin (ENBDM must equal one).
	Halt	Wait	Not supported.

[1] An analog connection from this pin to the on-chip regulator wakes up the regulator, which then initiates a power-on-reset sequence.

Individual power states are discussed in more detail in the following sections.

3.4   Debug Mode

Debug mode functions are managed through the background debug controller (BDC) in the Version 1 ColdFire core. The BDC provides the means for analyzing MCU operation during software development.

The debug interface is used to program a bootloader or user application program into the flash program memory before the MCU is operated in run mode for the first time. When the MCF51QE128/64/32 is shipped from the Freescale Semiconductor factory, the flash program memory is erased by default unless specifically noted, so there is no program that could be executed in run mode until the flash memory is initially programmed. The debug interface can also be used to erase and reprogram the flash memory after it has been previously programmed.

See Chapter 18, "Version 1 ColdFire Debug (CF1_DEBUG)," for more details regarding the debug interface.

3.5   Secure Mode

While the MCU is in secure mode, there are severe restrictions on which debug commands can be used. In this mode, only the upper byte of the core's XCSR, CSR2, and CSR3 registers can be accessed. See Chapter 18, "Version 1 ColdFire Debug (CF1_DEBUG)," for details.

3.6   Run Modes

3.6.1   Run Mode

Run mode is the normal operating mode for the MCF51QE128/64/32. This mode is selected when the BKGD/MS pin is high at the rising edge of the internal reset signal. Upon exiting reset, the CPU fetches the supervisor SR and initial PC from locations 0x(00)00_0000 and 0x(00)00_0004 in the memory map and executes code starting at the newly set value of the PC.

3.6.2   Low-Power Run Mode (LPrun)

In the low-power run mode, the on-chip voltage regulator is put into its standby (or loose regulation) state. In this state, the power consumption is reduced to a minimum that allows CPU functionality. Power consumption is reduced the most by disabling the clocks to all unused peripherals by clearing the corresponding bits in the SCGC1 and SCGC2 registers[1].

Before entering this mode, the following conditions must be met:

- FBELP[2] is the selected clock mode for the ICS. See Section 12.1.5.6, "FLL Bypassed External Low Power (FBELP)," for more details.
- ICSC2[HGO] is cleared.
- The bus frequency is less than 125 kHz.
- The ADC must be in low-power mode (ADLPC=1) or disabled.
- Low-voltage detect must be disabled. The LVDE and/or LVDSE bit in SPMSC1 register must be cleared.
- Flash programming/erasing is not allowed

After these conditions are met, low-power run mode can be entered by setting SPMSC2[LPR].

To re-enter standard run mode, clear the LPR bit. SPMSC2[LPRS] is a read-only status bit that can be used to determine if the regulator is in full-regulation mode or not. When LPRS is cleared, the regulator is in full-regulation mode and the MCU can run at full speed in any clock mode.

Assuming that SOPT1[BKGDPE] is set to enable BKGD/MS, the device also switches from LPrun to run mode when it detects a negative transition on the BKGD/MS pin.

Low-power run mode also provides the option to return to full regulation if any interrupt occurs. This is done by setting SPMSC2[LPWUI]. The ICS can then be set for full speed immediately in the interrupt service routine.

3.6.2.1   BDM in Low-Power Run Mode

Low-power run mode cannot be entered when the MCU is in active background debug mode.

If a device is in low-power run mode, a falling edge on an active BKGD/MS pin exits low-power run mode, clears the LPR and LPRS bits, and returns the device to normal run mode.

1. System clock gating control registers 1 and 2
2. FLL bypassed external low-power

3.7 Wait Modes

3.7.1 Wait Mode

Wait mode is entered by executing a STOP instruction after configuring the device as per Table 3-1. Upon execution of the STOP instruction, the CPU enters a low-power state in which it is not clocked.

The V1 ColdFire core does not differentiate between stop and wait modes. Both are stop from the core's perspective. The difference between the two is at the device level. In stop mode, most peripheral clocks are shut down. In wait mode, they continue to run.

XCSR[ENBDM] must be set prior to entering wait mode if the device is required to respond to BDM commands once in wait.

When an interrupt request occurs, the CPU exits wait mode and resumes with exception processing, beginning with the stacking operations leading to the interrupt service routine.

3.7.2 Low-Power Wait Mode (LPwait)

Low-power wait mode is entered by executing a STOP instruction while the MCU is in low-power run mode and configured per Table 3-1. In the low-power wait mode, the on-chip voltage regulator remains in its standby state as in the low-power run mode. In this state, the power consumption is reduced to a minimum that allows most modules to maintain funtionality. Power consumption is reduced the most by disabling the clocks to all unused peripherals by clearing the corresponding bits in the SCGC registers.

Low-power run mode restrictions also apply to low-power wait mode.

If the LPWUI bit is set when the STOP instruction is executed, the voltage regulator returns to full regulation when wait mode is exited. The ICS can be set for full speed immediately in the interrupt service routine.

If the LPWUI bit is cleared when the STOP instruction is executed, the device returns to low-power run mode.

Any reset exits low-power wait mode, clears the LPR bit, and returns the device to normal run mode.

3.7.2.1 BDM in Low-Power Wait Mode

If a device is in low-power wait mode, a falling edge on an active BKGD/MS pin exits low-power wait mode, clears the LPR and LPRS bits, and returns the device to normal run mode.

3.8 Stop Modes

One of three stop modes is entered upon execution of a STOP instruction when SOPT1[STOPE] is set. The SOPT1[WAITE] bit must be clear, else wait mode is entered. In stop3 mode, the bus and CPU clocks are halted. If the ENBDM bit is set prior to entering stop4, only the peripheral clocks are halted. The ICS module can be configured to leave the reference clocks running. See Chapter 12, "Internal Clock Source (S08ICSV3)," for more information.

The stop modes are selected by setting the appropriate bits in the system power management status and
control 2 (SPMSC2) register. Table 3-1 shows all of the control bits that affect mode selection under
various conditions. The selected mode is entered following the execution of a STOP instruction.

Most background commands are not available in stop mode. The memory-access-with-status commands
do not allow memory access, but they report an error indicating that the MCU is in either stop or wait
mode. The BACKGROUND command can be used to wake the MCU from stop4 and enter halt mode if
the ENBDM bit was set prior to entering stop. After entering halt mode, all background commands are
available.

3.8.1   Stop2 Mode

Stop2 mode is entered by executing a STOP instruction under the conditions as shown in Table 3-1.

Most of the internal circuitry of the MCU is powered off in stop2 with the exception of the RAM and
optionally the RTC. Upon entering stop2, all I/O pin control signals are latched so that the pins retain their
states during stop2.

Exit from stop2 is performed by driving the wake-up pin (PTA5/IRQ/TPM1CLK/$\overline{\text{RESET}}$) on the MCU to
zero.

NOTE

PTA5/IRQ/TPM1CLK/$\overline{\text{RESET}}$ functions as an active-low wakeup input
when the MCU is in stop2, as long as the pin is configured as an input before
entering stop2. The pullup on this pin is not automatically enabled in stop2.
To enable the internal pullup, set PTAPE[PTAPE5].

In addition, the real-time counter (RTC) can wake the MCU from stop2, if enabled.

Upon wake-up from stop2 mode, the MCU starts up as from a power-on reset (POR):

- All module control and status registers are reset, with the exception of the power management
  controller (SPMSC1/2/3), RTC, and debug trace buffer. Refer to the individual module chapters for
  more information on which other registers are unaffected by wake-up from stop2 mode.
- The LVD reset function is enabled and the MCU remains in the reset state if V_{DD} is below the LVD
  trip point (low trip point selected due to POR).
- The CPU initiates reset exception processing by fetching the vectors at 0x(00)00_0000 and
  0x(00)00_0004.

In addition to the above, upon waking up from stop2, SPMSC2[PPDF] is set. This flag is used to direct user code to go to a stop2 recovery routine. PPDF remains set and the I/O pin states remain latched until a 1 is written to SPMSC2[PPDACK].

Wakeup from stop2 can be initiated with an RTC interrupt. Unlike most other modules on the chip, the RTC is not reset as a result of exiting stop2. This implies that the RTC interrupt is asserted (although masked) upon exit from stop2.

To maintain I/O states for pins configured as general-purpose I/O before entering stop2, restore the contents of the I/O port registers, which have been saved in RAM, to the port registers before writing to the PPDACK bit. If the port registers are not restored from RAM before writing to PPDACK, the pins switch to their reset states when PPDACK is written.

For pins that were configured as peripheral I/O, reconfigure the peripheral module that interfaces to the pin before writing to PPDACK. If the peripheral module is not enabled before writing to PPDACK, the pins are controlled by their associated port control registers when the I/O latches are opened.

3.8.1.1   Low-Range Oscillator Considerations for Stop2

If using a low-range oscillator during stop2, reconfigure the ICSC2 register before PPDACK is written. The low-range oscillator (ICSC2[RANGE] = 0) can operate in stop2 as the clock source for the RTC module. If the low-range oscillator is active when entering stop2, it remains active in stop2 regardless of the value of ICSC2[EREFSTEN]. To disable the oscillator in stop2, switch the ICS into FBI or FEI mode before executing the STOP instruction.

3.8.2   Stop3 Mode

Stop3 mode is entered by executing a STOP instruction under the conditions as shown in Table 3-1. The states of all of the internal registers and logic, RAM contents, and I/O pin states are maintained. The on-chip regulator is placed in standby state.

Stop3 can be exited by asserting $\overline{\text{RESET}}$ or by an interrupt from one of the following sources: the RTC, ADC, ACMP, IRQ, SCI, or KBI.

If stop3 is exited by the $\overline{\text{RESET}}$ pin, the MCU is reset and operation resumes after taking the reset vector. Exit by one of the internal interrupt sources results in the MCU taking the appropriate interrupt vector.

3.8.3   Stop4: Low Voltage Detect or BDM Enabled in Stop Mode

Stop4 is differentiated from stop2 and stop3 in that the on-chip regulator is fully engaged.

Entry into halt mode from run mode is enabled if the XCSR[ENBDM] bit is set. This register is described in Chapter 18, "Version 1 ColdFire Debug (CF1_DEBUG)". If ENBDM is set when the CPU executes a STOP instruction, the system clocks to the background debug logic remain active when the MCU enters stop mode. Because of this, background debug communication remains possible. If you attempt to enter stop2 or stop3 with ENBDM set, the MCU enters stop4 instead (see Table 3-1 for details).

Stop4 is also entered if SPMSC1[LVDE, LVDSE] are set, enabling low voltage detect when the STOP instruction is executed. The LVD may only be used when the on-chip regulator is in full regulation mode. Thus, stop3 and stop2 modes are not compatible with use of the LVD.

The LVD system is capable of generating an interrupt or a reset when the supply voltage drops below the LVD voltage.

Stop4 can be exited by asserting $\overline{\text{RESET}}$ or by an interrupt from one of the following sources: the RTC, LVD, LVW, ADC, ACMP*x*, IRQ, SCI or the KBI.

3.9   On-Chip Peripheral Modules in Stop and Low-Power Modes

When the MCU enters any stop mode (wait not included), system clocks to the internal peripheral modules are stopped. Even in the exception case (ENBDM = 1), where clocks to the background debug logic continue to operate, clocks to the peripheral systems are halted to reduce power consumption. Refer to Section 3.8.1, "Stop2 Mode," and Section 3.8.2, "Stop3 Mode," for specific information on system behavior in stop modes.

When the MCU enters LPwait or LPrun modes, system clocks to the internal peripheral modules continue based on the settings of the clock gating control registers (SCGC1 and SCGC2).

Table 3-3 defines terms used in Table 3-4 to describe operation of components on the chip in the various low-power modes.

Table 3-3. Abbreviations used in Table 3-4

Voltage Regulator	Clocked[1]	Not Clocked
Full Regulation	FullOn	FullNoClk FullADACK[2]
Soft Regulation	SoftOn[3]	SoftNoClk Disabled SoftADACK[4]
Off	N/A	Off

[1] Subject to module enables and settings of System Clock Gating Control Registers 1 and 2 (SCGC1 and SCGC2).

[2] This ADC-specific mode defines the case where the device is fully regulated and the normal peripheral clock is stopped. In this case, the ADC can run using its internally generated asynchronous ADACK clock.

[3] Analog modules must be in their low-power mode when the device is operated in this state.

[4] This ADC-specific mode defines the case where the device is in soft regulation and the normal peripheral clock is stopped. In this case, the ADC can only be run using its low-power mode and internally generated asynchronous ADACK clock.

Table 3-4. Low-Power Mode Behavior

Peripheral	Mode					
	Stop2	Stop3	Stop4	LPwait	Wait	LPrun
CF1_CORE	Off	SoftNoClk	FullNoClk	SoftNoClk	FullNoClk	SoftOn
RAM	SoftNoClk	SoftNoClk	FullNoClk	SoftNoClk	FullNoClk	SoftOn

MCF51QE128 MCU Series Reference Manual, Rev. 3

Table 3-4. Low-Power Mode Behavior (continued)

Peripheral	Mode					
	Stop2	**Stop3**	**Stop4**	**LPwait**	**Wait**	**LPrun**
Flash	Off	SoftNoClk	FullNoClk	SoftNoClk	FullNoClk	SoftOn
Port I/O Registers	Off	SoftNoClk	FullNoClk	SoftOn	FullOn	SoftOn
ADC[1,2]	Off	SoftADACK (Wake Up)	FULLADACK (Wake Up)	SoftOn	FullOn	SoftOn
ACMPx	Off	SoftNoClk (Wake Up)	FullNoClk (Wake Up)	SoftOn	FullOn	SoftOn
BDC	Off	SoftOn	On	SoftOn	FullOn	SoftOn
COP	Off	SoftNoClk	FullNoClk	SoftOn	FullOn	SoftOn
Crystal Oscillator	RANGE=0 HGO=0	RANGE=0 HGO=0	All Modes	RANGE=0 HGO=0	All Modes	RANGE=0 HGO=0
ICS	Off	Stop or FBELP[3]	Stop or any mode	FBELP	Any mode	FBELP
IICx	Off	SoftNoClk	FullNoClk	SoftOn	FullOn	SoftOn
IRQ	Off (Wake Up via POR)[4]	SoftNoClk (Wake Up)	FullNoClk (Wake Up)	SoftOn	FullOn	SoftOn
KBIx	Off	SoftNoClk (Wake Up)	FullNoClk (Wake Up)	SoftOn	FullOn	SoftOn
LVD/LVW	Off	Disabled	On (Wake Up)	Disabled	FullOn	Disabled
RTC	Soft Regulation, LPOCLK if enabled (Wake Up via POR)	SoftOn (Wake Up)	Full Regulation LPOCLK, ICSERCLK or ICSIRCLK only (Wake Up)	SoftOn	FullOn	SoftOn
SCIx	Off	SoftNoClk	FullNoClk	SoftOn	FullOn	SoftOn
SPIx	Off	SoftNoClk	FullNoClk	SoftOn	FullOn	SoftOn
TPMx	Off	SoftNoClk	FullNoClk	SoftOn	FullOn	SoftOn
Voltage Regulator / PMC	Parital Shutdown. 1 kHz osc if enabled	Soft Regulation. 1 kHz osc if enabled	Full Regulation 1 kHz osc on	SoftOn 1 kHz osc on	FullOn 1 kHz osc on	SoftOn 1 kHz osc on
LPI/O Pins	States Held	SoftNoClk	FullNoClk	SoftOn	FullOn	SoftOn

[1] LP mode for the ADC is invoked by setting ADLPC=1. ADACK is selected via the ADCCFG[ADICLK] field in the ADC. See Chapter 11, "Analog-to-Digital Converter (S08ADC12V1)," for details.

[2] LVD must be enabled to run in stop if converting the bandgap channel.

[3] FBELP refers to the ICS FLL bypassed external low-power state. See Chapter 12, "Internal Clock Source (S08ICSV3)," for more details.

[4] The PTA5/IRQ/TPM1CLK/$\overline{\text{RESET}}$ pin also has a direct connection to the on-chip regulator wakeup input. Asserting this pin low while in stop2 triggers the PMC to wakeup. As a result, the device undergoes a power-on-reset sequence.

Chapter 4
Memory

4.1   MCF51QE128/64/32 Memory Map

As shown in Figure 4-1, on-chip memory in the MCF51QE128/64/32 series of MCUs consists of RAM and flash program memory for nonvolatile code and data storage, plus I/O and control/status registers.

NOTE

Version 1 ColdFire devices contain 24-bit internal address buses, while previous ColdFire cores have 32-bit internal address buses. Because there may be some resources that use a 32-bit address, this chapter shows 24-bit and 32-bit addresses by indicating the extra upper byte in parentheses.

MCF51QE128

CPU Address

Address	Region
0x(00)00_0000	Flash **128 Kbytes**
0x(00)01_FFFF 0x(00)02_0000	Unimplemented
0x(00)7F_FFFF 0x(00)80_0000	RAM **8 Kbytes**
0x(00)80_1FFF 0x(00)80_2000	Unimplemented
0x(00)BF_FFFF 0x(00)C0_0000	ColdFire Rapid GPIO
0x(00)C0_000F 0x(00)C0_0010	Unimplemented
0x(FF)FF_7FFF 0x(FF)FF_8000	Slave Peripherals
0x(FF)FF_FFFF	

MCF51QE64

CPU Address

Address	Region
0x(00)00_0000 0x(00)00_FFFF	Flash **64 Kbytes**
0x(00)01_0000	Unimplemented
0x(00)7F_FFFF 0x(00)80_0000	RAM **8 Kbytes**
0x(00)80_1FFF 0x(00)80_2000	Unimplemented
0x(00)BF_FFFF 0x(00)C0_0000	ColdFire Rapid GPIO
0x(00)C0_000F 0x(00)C0_0010	Unimplemented
0x(FF)FF_7FFF 0x(FF)FF_8000	Slave Peripherals
0x(FF)FF_FFFF	

MCF51QE32

CPU Address

Address	Region
0x(00)00_0000 0x(00)00_7FFF 0x(00)00_8000	Flash **32 Kbytes**
	Unimplemented
0x(00)7F_FFFF 0x(00)80_0000	RAM **8 Kbytes**
0x(00)80_1FFF 0x(00)80_2000	Unimplemented
0x(00)BF_FFFF 0x(00)C0_0000	ColdFire Rapid GPIO
0x(00)C0_000F 0x(00)C0_0010	Unimplemented
0x(FF)FF_7FFF 0x(FF)FF_8000	Slave Peripherals
0x(FF)FF_FFFF	

Figure 4-1. MCF51QE128/64/32 Memory Maps

Regions within the memory map are subject to restrictions with regard to the types of allowable accesses. These are outlined in Table 4-1. Non-supported access types terminate the bus cycle with an error and would typically generate a system reset in response to the error termination.

Table 4-1. CPU Access Type Allowed by Region

Base Address	Region	Read			Write		
		Byte	Word	Long	Byte	Word	Long
0x(00)00_0000	Flash	x	x	x	—	—	x
0x(00)80_0000	RAM	x	x	x	x	x	x
0x(00)C0_0000	Rapid GPIO	x	x	x	x	x	x
0x(FF)FF_8000	Peripherals	x	x	—	x	x	—

Consistent with past ColdFire devices, flash configuration data is located at 0x400.

The slave peripherals section of the memory map is further broken into the following sub-sections:

```
0x(FF)FF_8000 – 0x(FF)FF_807F        Direct-page peripheral regs
0x(FF)FF_9800 – 0x(FF)FF_98FF        High-page peripheral regs
0x(FF)FF_FFC0 – 0x(FF)FF_FFFC        Interrupt controller
```

The section of memory at 0x(00)C0_0000 is assigned for use by the ColdFire Rapid GPIO module. See Section 4.2.2, "ColdFire Rapid GPIO Memory Map," for the rapid GPIO memory map and Section 6.4, "V1 ColdFire Rapid GPIO Functionality," for further details on the module.

The MCF51QE128/64/32 devices utilize an 8-bit peripheral bus. The bus bridge from the ColdFire system bus to the peripheral bus is capable of serializing 16-bit accesses into two 8-bit accesses. This can be used to speed access to properly aligned peripheral registers. Note, not all peripheral registers are aligned to take advantage of this feature.

CPU accesses to those parts of the memory map marked as unimplemented in Figure 4-1 result in an illegal address reset if CPUCR[ARD] is cleared or an address error exception if CPUCR[ARD] is set.

The lower 32K of flash memory and slave peripherals sections of the memory map are most efficiently accessed using the ColdFire absolute short addressing mode. RAM is most efficiently accessed using the A5-relative addressing mode (address register indirect with displacement mode).

4.2   Register Addresses and Bit Assignments

Peripheral registers in the MCF51QE128/64/32 are divided into two groups:

- Direct-page registers are located at 0x(FF)FF_8000 in the memory map.
- High-page registers are located at 0x(FF)FF_9800 in the memory map.

  There is no functional advantage to locating peripherals in the direct page versus the high page peripheral space for the MCF51QE128/64/32. Both sets of registers may be efficiently accessed using the ColdFire absolute short addressing mode. The areas are differentiated to maintain documentation compatibility with the MC9S08QE128/64/32, where there are significant performance issues.

NOTE

Peripheral register locations for MCF51QE128/64/32 are shifted 0x(FF)FF_8000 compared with the MC9S08QE128/64/32 devices.

- The ColdFire interrupt controller module is mapped in the peripheral space and occupies a 64-byte space at the upper end of memory. Accordingly, its address decode is defined as 0x(FF)FF_FFC0–0x(FF)FF_FFFF. This 64-byte space includes the program-visible registers as well as the space used for interrupt acknowledge (IACK) cycles.

- There is a nonvolatile register area consisting of a 16-byte block locted in flash memory at 0x(00)00_0400–0x(00)00_040F. Nonvolatile register locations include:
  - — NVPROT and NVOPT are loaded into working registers at reset
  - — An 8-byte backdoor comparison key that optionally allows a user to gain controlled access to secure memory

  Because the nonvolatile register locations are flash memory, they must be erased and programmed like other flash memory locations.

Table 4-2 is a summary of all user-accessible direct-page registers and control bits.

The register names in column two in Table 4-2, Table 4-3, Table 4-6, and Table 4-7 are shown in bold to set them apart from the bit names to the right. Cells not associated with named bits are shaded. A shaded cell with a 0 indicates this unused bit always reads as a 0. Shaded cells with dashes indicate unused or reserved bit locations that could read as 1s or 0s. When writing to these bits, write a 0 unless otherwise specified.

Recall that ColdFire uses a big-endian, byte-addressable memory architecture. The most significant byte of each address is the lowest numbered as shown in Figure 4-2. Multi-byte operands (16-bit words and 32-bit longwords) are referenced using an address pointing to the most significant (first) byte.

31	24	23	16	15	8	7	0
Longword 0x(00)00_0000							
Word 0x(00)00_0000				Word 0x(00)00_0002			
Byte 0x(00)00_0000		Byte 0x(00)00_0001		Byte 0x(00)00_0002		Byte 0x(00)00_0003	
Longword 0x(00)00_0004							
Word 0x(00)00_0004				Word 0x(00)00_0006			
Byte 0x(00)00_0004		Byte 0x(00)00_0005		Byte 0x(00)00_0006		Byte 0x(00)00_0007	
• • •							
Longword 0x(FF)FF_FFFC							
Word 0x(FF)FF_FFFC				Word 0x(FF)FF_FFFE			
Byte 0x(FF)FF_FFFC		Byte 0x(FF)FF_FFFD		Byte 0x(FF)FF_FFFE		Byte 0x(FF)FF_FFFF	

Figure 4-2. ColdFire Memory Organization

Table 4-2. Direct-Page Register Summary (Sheet 1 of 4)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8000	**PTAD**	PTAD7	PTAD6	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
0x(FF)FF_8001	**PTADD**	PTADD7	PTADD6	PTADD5	PTADD4	PTADD3	PTADD2	PTADD1	PTADD0
0x(FF)FF_8002	**PTBD**	PTBD7	PTBD6	PTBD5	PTBD4	PTBD3	PTBD2	PTBD1	PTBD0
0x(FF)FF_8003	**PTBDD**	PTBDD7	PTBDD6	PTBDD5	PTBDD4	PTBDD3	PTBDD2	PTBDD1	PTBDD0
0x(FF)FF_8004	**PTCD**	PTCD7	PTCD6	PTCD5	PTCD4	PTCD3	PTCD2	PTCD1	PTCD0
0x(FF)FF_8005	**PTCDD**	PTCDD7	PTCDD6	PTCDD5	PTCDD4	PTCDD3	PTCDD2	PTCDD1	PTCDD0
0x(FF)FF_8006	**PTDD**	PTDD7	PTDD6	PTDD5	PTDD4	PTDD3	PTDD2	PTDD1	PTDD0
0x(FF)FF_8007	**PTDDD**	PTDDD7	PTDDD6	PTDDD5	PTDDD4	PTDDD3	PTDDD2	PTDDD1	PTDDD0
0x(FF)FF_8008	**PTED**	PTED7	PTED6	PTED5	PTED4	PTED3	PTED2	PTED1	PTED0
0x(FF)FF_8009	**PTEDD**	PTEDD7	PTEDD6	PTEDD5	PTEDD4	PTEDD3	PTEDD2	PTEDD1	PTEDD0
0x(FF)FF_800A	**PTFD**	PTFD7	PTFD6	PTFD5	PTFD4	PTFD3	PTFD2	PTFD1	PTFD0
0x(FF)FF_800B	**PTFDD**	PTFDD7	PTFDD6	PTFDD5	PTFDD4	PTFDD3	PTFDD2	PTFDD1	PTFDD0
0x(FF)FF_800C	**KBI1SC**	0	0	0	0	KBF	KBACK	KBIE	KBIMOD
0x(FF)FF_800D	**KBI1PE**	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
0x(FF)FF_800E	**KBI1ES**	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBEDG3	KBEDG2	KBEDG1	KBEDG0
0x(FF)FF_800F	**IRQSC**	0	IRQPDD	IRQEDG	IRQPE	IRQF	IRQACK	IRQIE	IRQMOD
0x(FF)FF_8010	**ADSC1**	COCO	AIEN	ADCO	ADCH				
0x(FF)FF_8011	**ADSC2**	ADACT	ADTRG	ACFE	ACFGT	0	0	0	0
0x(FF)FF_8012	**ADRH**	0	0	0	0	0	0	ADR9	ADR8
0x(FF)FF_8013	**ADRL**	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0
0x(FF)FF_8014	**ADCVH**	0	0	0	0	0	0	ADCV9	ADCV8
0x(FF)FF_8015	**ADCVL**	ADCV7	ADCV6	ADCV5	ADCV4	ADCV3	ADCV2	ADCV1	ADCV0
0x(FF)FF_8016	**ADCFG**	ADLPC	ADIV		ADLSMP	MODE		ADICLK	
0x(FF)FF_8017	**APCTL1**	ADPC7	ADPC6	ADPC5	ADPC4	ADPC3	ADPC2	ADPC1	ADPC0
0x(FF)FF_8018	**APCTL2**	ADPC15	ADPC14	ADPC13	ADPC12	ADPC11	ADPC10	ADPC9	ADPC8
0x(FF)FF_8019	**APCTL3**	ADPC23	ADPC22	ADPC21	ADPC20	ADPC19	ADPC18	ADPC17	ADPC16
0x(FF)FF_801A	**ACMP1SC**	ACME	ACBGS	ACF	ACIE	ACO	ACOPE	ACMOD1	ACMOD0
0x(FF)FF_801B	**ACMP2SC**	ACME	ACBGS	ACF	ACIE	ACO	ACOPE	ACMOD1	ACMOD0
0x(FF)FF_801C	**PTGD**	PTGD7	PTGD6	PTGD5	PTGD4	PTGD3	PTGD2	PTGD1	PTGD0
0x(FF)FF_801D	**PTGDD**	PTGDD7	PTGDD6	PTGDD5	PTGDD4	PTGDD3	PTGDD2	PTGDD1	PTGDD0
0x(FF)FF_801E	**PTHD**	PTHD7	PTHD6	PTHD5	PTHD4	PTHD3	PTHD2	PTHD1	PTHD0
0x(FF)FF_801F	**PTHDD**	PTHDD7	PTHDD6	PTHDD5	PTHDD4	PTHDD3	PTHDD2	PTHDD1	PTHDD0

MCF51QE128 MCU Series Reference Manual, Rev. 3

Freescale Semiconductor

Table 4-2. Direct-Page Register Summary (Sheet 2 of 4)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8020	**SCI1BDH**	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8
0x(FF)FF_8021	**SCI1BDL**	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
0x(FF)FF_8022	**SCI1C1**	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
0x(FF)FF_8023	**SCI1C2**	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0x(FF)FF_8024	**SCI1S1**	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
0x(FF)FF_8025	**SCI1S2**	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
0x(FF)FF_8026	**SCI1C3**	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
0x(FF)FF_8027	**SCI1D**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8028	**SPI1C1**	SPI1E	SP1E	SP1TIE	MSTR	CPOL	CPHA	SSOE	LSBFE
0x(FF)FF_8029	**SPI1C2**	0	0	0	MODFEN	BIDIROE	0	SPI1SWAI	SP1C0
0x(FF)FF_802A	**SPI1BR**	0	SP1PR2	SP1PR1	SP1PR0	0	SP1R2	SP1R1	SP1R0
0x(FF)FF_802B	**SPI1S**	SP1RF	0	SP1TEF	MODF	0	0	0	0
0x(FF)FF_802C	**Reserved**	0	0	0	0	0	0	0	0
0x(FF)FF_802D	**SPI1D**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_802E	**PTJD**	PTJD7	PTJD6	PTJD5	PTJD4	PTJD3	PTJD2	PTJD1	PTJD0
0x(FF)FF_802F	**PTJDD**	PTJDD7	PTJDD6	PTJDD5	PTJDD4	PTJDD3	PTJDD2	PTJDD1	PTJDD0
0x(FF)FF_8030	**IIC1A**	AD7	AD6	AD5	AD4	AD3	AD2	AD1	0
0x(FF)FF_8031	**IIC1F**	MULT		TAP2			TAP1		
0x(FF)FF_8032	**IIC1C1**	IICEN	IICIE	MST	TX	TXAK	RSTA	0	0
0x(FF)FF_8033	**IIC1S**	TCF	IAAS	BUSY	ARBL	0	SRW	IICIF	RXAK
0x(FF)FF_8034	**IIC1D**	DATA							
0x(FF)FF_8035	**IIC1C2**	GCAEN	ADEXT	0	0	0	AD10	AD9	AD8
0x(FF)FF_8036	**Reserved**	—	—	—	—	—	—	—	—
0x(FF)FF_8037	**ICSTEST**	RESERVED FOR FACTORY USE							
0x(FF)FF_8038	**ICSC1**	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
0x(FF)FF_8039	**ICSC2**	BDIV		RANGE	HGO	LP	EREFS	ERCLKEN	EREFSTEN
0x(FF)FF_803A	**ICSTRM**	TRIM							
0x(FF)FF_803B	**ICSSC**	DFR		DMX32	IREFST	CLKST		OSCINIT	FTRIM
0x(FF)FF_803C	**KBI2SC**	0	0	0	0	KBF	KBACK	KBIE	KBIMOD
0x(FF)FF_803D	**KBI2PE**	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
0x(FF)FF_803E	**KBI2ES**	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBEDG3	KBEDG2	KBEDG1	KBEDG0
0x(FF)FF_803F	**Reserved**	—	—	—	—	—	—	—	—

MCF51QE128 MCU Series Reference Manual, Rev. 3

Get the latest version from freescale.com

Table 4-2. Direct-Page Register Summary (Sheet 3 of 4)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8040	**TPM1SC**	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
0x(FF)FF_8041	**TPM1CNTH**	Bit 15	14	13	12	11	10	9	Bit 8
0x(FF)FF_8042	**TPM1CNTL**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8043	**TPM1MODH**	Bit 15	14	13	12	11	10	9	Bit 8
0x(FF)FF_8044	**TPM1MODL**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8045	**TPM1C0SC**	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x(FF)FF_8046	**TPM1C0VH**	Bit 15	14	13	12	11	10	9	Bit 8
0x(FF)FF_8047	**TPM1C0VL**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8048	**TPM1C1SC**	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
0x(FF)FF_8049	**TPM1C1VH**	Bit 15	14	13	12	11	10	9	Bit 8
0x(FF)FF_804A	**TPM1C1VL**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_804B	**TPM1C2SC**	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	0	0
0x(FF)FF_804C	**TPM1C2VH**	Bit 15	14	13	12	11	10	9	Bit 8
0x(FF)FF_804D	**TPM1C2VL**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_804E–0x(FF)FF_804F	**Reserved**	—	—	—	—	—	—	—	—
0x(FF)FF_8050	**TPM2SC**	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
0x(FF)FF_8051	**TPM2CNTH**	Bit 15	14	13	12	11	10	9	Bit 8
0x(FF)FF_8052	**TPM2CNTL**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8053	**TPM2MODH**	Bit 15	14	13	12	11	10	9	Bit 8
0x(FF)FF_8054	**TPM2MODL**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8055	**TPM2C0SC**	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x(FF)FF_8056	**TPM2C0VH**	Bit 15	14	13	12	11	10	9	Bit 8
0x(FF)FF_8057	**TPM2C0VL**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8058	**TPM2C1SC**	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
0x(FF)FF_8059	**TPM2C1VH**	Bit 15	14	13	12	11	10	9	Bit 8
0x(FF)FF_805A	**TPM2C1VL**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_805B	**TPM2C2SC**	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	0	0
0x(FF)FF_805C	**TPM2C2VH**	Bit 15	14	13	12	11	10	9	Bit 8
0x(FF)FF_805D	**TPM2C2VL**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_805E–0x(FF)FF_805F	**Reserved**	—	—	—	—	—	—	—	—
0x(FF)FF_8060	**TPM3SC**	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0

MCF51QE128 MCU Series Reference Manual, Rev. 3

Table 4-2. Direct-Page Register Summary (Sheet 4 of 4)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8061	**TPM3CNTH**	Bit 15	14	13	12	11	10	9	Bit 8
0x(FF)FF_8062	**TPM3CNTL**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8063	**TPM3MODH**	Bit 15	14	13	12	11	10	9	Bit 8
0x(FF)FF_8064	**TPM3MODL**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8065	**TPM3C0SC**	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x(FF)FF_8066	**TPM3C0VH**	Bit 15	14	13	12	11	10	9	Bit 8
0x(FF)FF_8067	**TPM3C0VL**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8068	**TPM3C1SC**	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
0x(FF)FF_8069	**TPM3C1VH**	Bit 15	14	13	12	11	10	9	Bit 8
0x(FF)FF_806A	**TPM3C1VL**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_806B	**TPM3C2SC**	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	0	0
0x(FF)FF_806C	**TPM3C2VH**	Bit 15	14	13	12	11	10	9	Bit 8
0x(FF)FF_806D	**TPM3C2VL**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_806E	**TPM3C3SC**	CH3F	CH3IE	MS3B	MS3A	ELS3B	ELS3A	0	0
0x(FF)FF_806F	**TPM3C3VH**	Bit 15	14	13	12	11	10	9	Bit 8
0x(FF)FF_8070	**TPM3C3VL**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8071	**TPM3C4SC**	CH4F	CH4IE	MS4B	MS4A	ELS4B	ELS4A	0	0
0x(FF)FF_8072	**TPM3C4VH**	Bit 15	14	13	12	11	10	9	Bit 8
0x(FF)FF_8073	**TPM3C4VL**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8074	**TPM3C5SC**	CH5F	CH5IE	MS5B	MS5A	ELS5B	ELS5A	0	0
0x(FF)FF_8075	**TPM3C5VH**	Bit 15	14	13	12	11	10	9	Bit 8
0x(FF)FF_8076	**TPM3C5VL**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_8077- 0x(FF)FF_807F	**Reserved**	—	—	—	—	—	—	—	—

Table 4-3. High-Page Register Summary (Sheet 1 of 5)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_9800	**SRS**	POR	PIN	COP	ILOP	ILAD	0	LVD	0
0x(FF)FF_9801	**Reserved**	0	0	0	0	0	0	0	0
0x(FF)FF_9802	**SOPT1**	COPE	COPT	STOPE	WAITE	0	RSTOPE	BKGDPE	RSTPE
0x(FF)FF_9803	**SOPT2**	COPCLKS	0	0	0	SPI1PS	ACIC2	IICPS	ACIC1

Table 4-3. High-Page Register Summary (Sheet 2 of 5)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_9804	**SIMCTSC (Reserved)**	TMODE				TEST	TRSTPE	TC	
0x(FF)FF_9805	**SIMCO (Reserved)**	0	0	0	0	0	CSS	PCOE	0
0x(FF)FF_9806	**SDIDH**	0	0	0	0	ID[11:8]			
0x(FF)FF_9807	**SDIDL**	ID[7:0]							
0x(FF)FF_9808	**SPMSC1**	LVDF	LVDACK	LVDIE	LVDRE	LVDSE	LVDE	0	BGBE
0x(FF)FF_9809	**SPMSC2**	LPR	LPRS	LPWUI	0	PPDF	PPDACK	PPDE	PPDC
0x(FF)FF_980A	**Reserved**	—	—	—	—	—	—	—	—
0x(FF)FF_980B	**SPMSC3**	LVWF	LVWACK	LVDV	LVWV	LVWIE	0	0	0
0x(FF)FF_980C-0x(FF)FF_980D	**Reserved**	—	—	—	—	—	—	—	—
0x(FF)FF_980E	**SCGC1**	TPM3	TPM2	TPM1	ADC	IIC2	IIC1	SCI2	SCI1
0x(FF)FF_980F	**SCGC2**	1	FLS	IRQ	KBI*x*	ACMP*x*	RTC	SPI2	SPI1
0x(FF)FF_9810-0x(FF)FF_981F	**Reserved**	—	—	—	—	—	—	—	—
0x(FF)FF_9820	**FCDIV**	FDIVLD	PRDIV8	FDIV					
0x(FF)FF_9821	**FOPT**	KEYEN		0	0	0	0	SEC	
0x(FF)FF_9822	**FRSV0 (Reserved)**	0	0	0	0	0	0	0	0
0x(FF)FF_9823	**FCNFG**	0	0	KEYACC	0	0	0	0	0
0x(FF)FF_9824	**FPROT**	FPS							FPOPEN
0x(FF)FF_9825	**FSTAT**	FCBEF	FCCF	FPVIOL	FACCERR	0	FBLANK	0	0
0x(FF)FF_9826	**FCMD**	0	FCMD						
0x(FF)FF_9827	**FRSV1 (Reserved)**	0	0	0	0	0	0	0	0
0x(FF)FF_9828	**FADDRHI (Reserved)**	0	0	0	0	0	0	0	0
0x(FF)FF_9829	**FADDRLO (Reserved)**	0	0	0	0	0	0	0	0
0x(FF)FF_982A	**FRSV2 (Reserved)**	0	0	0	0	0	0	0	0
0x(FF)FF_982B	**FRSV3 (Reserved)**	0	0	0	0	0	0	0	0
0x(FF)FF_982C	**FDATAHI1 (Reserved)**	0	0	0	0	0	0	0	0

Table 4-3. High-Page Register Summary (Sheet 3 of 5)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_982D	FDATALO1 (Reserved)	0	0	0	0	0	0	0	0
0x(FF)FF_982E	FDATAHI0 (Reserved)	0	0	0	0	0	0	0	0
0x(FF)FF_982F	FDATALO0 (Reserved)	0	0	0	0	0	0	0	0
0x(FF)FF_9830	RTCSC	RTIF	RTCLKS		RTIE	RTCPS			
0x(FF)FF_9831	RTCCNT	RTCCNT							
0x(FF)FF_9832	RTCMOD	RTCMOD							
0x(FF)FF_9833-0x(FF)FF_9837	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_9838	SPI2C1	SPI2E	SP2E	SP2TIE	MSTR	CPOL	CPHA	SSOE	LSBFE
0x(FF)FF_9839	SPI2C2	0	0	0	MODFEN	BIDIROE	0	SPI2SWAI	SP2C0
0x(FF)FF_983A	SPI2BR	0	SP2PR2	SP2PR1	SP2PR0	0	SP2R2	SP2R1	SP2R0
0x(FF)FF_983B	SPI2S	SP2RF	0	SP2TEF	MODF	0	0	0	0
0x(FF)FF_983C	Reserved	0	0	0	0	0	0	0	0
0x(FF)FF_983D	SPI2D	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_983E-0x(FF)FF_983F	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_9840	PTAPE	PTAPE7	PTAPE6	PTAPE5	PTAPE4	PTAPE3	PTAPE2	PTAPE1	PTAPE0
0x(FF)FF_9841	PTASE	PTASE7	PTASE6	PTASE5	PTASE4	PTASE3	PTASE2	PTASE1	PTASE0
0x(FF)FF_9842	PTADS	PTADS7	PTADS6	PTADS5	PTADS4	PTADS3	PTADS2	PTADS1	PTADS0
0x(FF)FF_9843	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_9844	PTBPE	PTBPE7	PTBPE6	PTBPE5	PTBPE4	PTBPE3	PTBPE2	PTBPE1	PTBPE0
0x(FF)FF_9845	PTBSE	PTBSE7	PTBSE6	PTBSE5	PTBSE4	PTBSE3	PTBSE2	PTBSE1	PTBSE0
0x(FF)FF_9846	PTBDS	PTBDS7	PTBDS6	PTBDS5	PTBDS4	PTBDS3	PTBDS2	PTBDS1	PTBDS0
0x(FF)FF_9847	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_9848	PTCPE	PTCPE7	PTCPE6	PTCPE5	PTCPE4	PTCPE3	PTCPE2	PTCPE1	PTCPE0
0x(FF)FF_9849	PTCSE	PTCSE7	PTCSE6	PTCSE5	PTCSE4	PTCSE3	PTCSE2	PTCSE1	PTCSE0
0x(FF)FF_984A	PTCDS	PTCDS7	PTCDS6	PTCDS5	PTCDS4	PTCDS3	PTCDS2	PTCDS1	PTCDS0
0x(FF)FF_984B	Reserved	—	—	—	—	—	—	—	—

MCF51QE128 MCU Series Reference Manual, Rev. 3

Table 4-3. High-Page Register Summary (Sheet 4 of 5)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_984C	**PTDPE**	PTDPE7	PTDPE6	PTDPE5	PTDPE4	PTDPE3	PTDPE2	PTDPE1	PTDPE0
0x(FF)FF_984D	**PTDSE**	PTDSE7	PTDSE6	PTDSE5	PTDSE4	PTDSE3	PTDSE2	PTDSE1	PTDSE0
0x(FF)FF_984E	**PTDDS**	PTDDS7	PTDDS6	PTDDS5	PTDDS4	PTDDS3	PTDDS2	PTDDS1	PTDDS0
0x(FF)FF_984F	**Reserved**	—	—	—	—	—	—	—	—
0x(FF)FF_9850	**PTEPE**	PTEPE7	PTEPE6	PTEPE5	PTEPE4	PTEPE3	PTEPE2	PTEPE1	PTEPE0
0x(FF)FF_9851	**PTESE**	PTESE7	PTESE6	PTESE5	PTESE4	PTESE3	PTESE2	PTESE1	PTESE0
0x(FF)FF_9852	**PTEDS**	PTEDS7	PTEDS6	PTEDS5	PTEDS4	PTEDS3	PTEDS2	PTEDS1	PTEDS0
0x(FF)FF_9853	**Reserved**	—	—	—	—	—	—	—	—
0x(FF)FF_9854	**PTFPE**	PTFPE7	PTFPE6	PTFPE5	PTFPE4	PTFPE3	PTFPE2	PTFPE1	PTFPE0
0x(FF)FF_9855	**PTFSE**	PTFSE7	PTFSE6	PTFSE5	PTFSE4	PTFSE3	PTFSE2	PTFSE1	PTFSE0
0x(FF)FF_9856	**PTFDS**	PTFDS7	PTFDS6	PTFDS5	PTFDS4	PTFDS3	PTFDS2	PTFDS1	PTFDS0
0x(FF)FF_9857	**Reserved**	—	—	—	—	—	—	—	—
0x(FF)FF_9858	**PTGPE**	PTGPE7	PTGPE6	PTGPE5	PTGPE4	PTGPE3	PTGPE2	PTGPE1	PTGPE0
0x(FF)FF_9859	**PTGSE**	PTGSE7	PTGSE6	PTGSE5	PTGSE4	PTGSE3	PTGSE2	PTGSE1	PTGSE0
0x(FF)FF_985A	**PTGDS**	PTGDS7	PTGDS6	PTGDS5	PTGDS4	PTGDS3	PTGDS2	PTGDS1	PTGDS0
0x(FF)FF_985B	**Reserved**	—	—	—	—	—	—	—	—
0x(FF)FF_985C	**PTHPE**	PTHPE7	PTHPE6	PTHPE5	PTHPE4	PTHPE3	PTHPE2	PTHPE1	PTHPE0
0x(FF)FF_985D	**PTHSE**	PTHSE7	PTHSE6	PTHSE5	PTHSE4	PTHSE3	PTHSE2	PTHSE1	PTHSE0
0x(FF)FF_985E	**PTHDS**	PTHDS7	PTHDS6	PTHDS5	PTHDS4	PTHDS3	PTHDS2	PTHDS1	PTHDS0
0x(FF)FF_985F	**Reserved**	—	—	—	—	—	—	—	—
0x(FF)FF_9860	**PTJPE**	PTJPE7	PTJPE6	PTJPE5	PTJPE4	PTJPE3	PTJPE2	PTJPE1	PTJPE0
0x(FF)FF_9861	**PTJSE**	PTJSE7	PTJSE6	PTJSE5	PTJSE4	PTJSE3	PTJSE2	PTJSE1	PTJSE0
0x(FF)FF_9862	**PTJDS**	PTJDS7	PTJDS6	PTJDS5	PTJDS4	PTJDS3	PTJDS2	PTJDS1	PTJDS0
0x(FF)FF_9863-0x(FF)FF_9867	**Reserved**	—	—	—	—	—	—	—	—
0x(FF)FF_9868	**IIC2A**	AD7	AD6	AD5	AD4	AD3	AD2	AD1	0
0x(FF)FF_9869	**IIC2F**	MULT		TAP2			TAP1		
0x(FF)FF_986A	**IIC2C1**	IICEN	IICIE	MST	TX	TXAK	RSTA	0	0
0x(FF)FF_986B	**IIC2S**	TCF	IAAS	BUSY	ARBL	0	SRW	IICIF	RXAK
0x(FF)FF_986C	**IIC2D**	DATA							
0x(FF)FF_986D	**IIC2C2**	GCAEN	ADEXT	0	0	0	AD10	AD9	AD8
0x(FF)FF_986E-0x(FF)FF_986F	**Reserved**	—	—	—	—	—	—	—	—

MCF51QE128 MCU Series Reference Manual, Rev. 3

Freescale Semiconductor

Table 4-3. High-Page Register Summary (Sheet 5 of 5)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_9870	**SCI2BDH**	LBKDIE	RXEDGIE	0	SBR[12:8]				
0x(FF)FF_9871	**SCI2BDL**	SBR[7:0]							
0x(FF)FF_9872	**SCI2C1**	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
0x(FF)FF_9873	**SCI2C2**	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0x(FF)FF_9874	**SCI2S1**	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
0x(FF)FF_9875	**SCI2S2**	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
0x(FF)FF_9876	**SCI2C3**	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
0x(FF)FF_9877	**SCI2D**	Bit 7	6	5	4	3	2	1	Bit 0
0x(FF)FF_9878	**PTCSET**	PTCSET7	PTCSET6	PTCSET5	PTCSET4	PTCSET3	PTCSET2	PTCSET1	PTCSET0
0x(FF)FF_9879	**PTESET**	PTESET7	PTESET6	PTESET5	PTESET4	PTESET3	PTESET2	PTESET1	PTESET0
0x(FF)FF_987A	**PTCCLR**	PTCCLR7	PTCCLR6	PTCCLR5	PTCCLR4	PTCCLR3	PTCCLR2	PTCCLR1	PTCCLR0
0x(FF)FF_987B	**PTECLR**	PTECLR7	PTECLR6	PTECLR5	PTECLR4	PTECLR3	PTECLR2	PTECLR1	PTECLR0
0x(FF)FF_987C	**PTCTOG**	PTCTOG7	PTCTOG6	PTCTOG5	PTCTOG4	PTCTOG3	PTCTOG2	PTCTOG1	PTCTOG0
0x(FF)FF_987D	**PTETOG**	PTETOG7	PTETOG6	PTETOG5	PTETOG4	PTETOG3	PTETOG2	PTETOG1	PTETOG0
0x(FF)FF_987E- 0x(FF)FF_987F	**Reserved**	—	—	—	—	—	—	—	—

4.2.1 Flash Module Reserved Memory Locations

Several reserved flash memory locations, shown in Table 4-4, are used for storing values used by corresponding peripheral registers. These registers include an 8-byte backdoor key that can be used to gain access to secure memory resources. During reset events, the contents of the flash protection byte (NVPROT) and flash nonvolatile byte (NVOPT) in the reserved flash memory are transferred into the corresponding FPROT and FOPT registers in the high-page register area to control security and block protection options.

Table 4-4. Reserved Flash Memory Addresses

Address	MSB[1] (0x0)	(0x1)	(0x2)	LSB[2] (0x3)
0x(00)00_03FC	Reserved		FTRIM (bit 0)	TRIM
0x(00)00_0400	Backdoor comparison key bytes 0–3			
	byte0	byte1	byte2	byte3
0x(00)00_0404	Backdoor comparison key bytes 4–7			
	byte4	byte5	byte6	byte7

Table 4-4. Reserved Flash Memory Addresses

Address	MSB[1] (0x0)	(0x1)	(0x2)	LSB[2] (0x3)
0x(00)00_0408	Reserved			
0x(00)00_040C	Reserved	NVPROT	Reserved	NVOPT

[1] MSB = most significant byte

[2] LSB = least significant byte

Table 4-5. Reserved Flash Memory Addresses

Address	Register	7	6	5	4	3	2	1	0
0x(00)00_03FC– 0x(00)00_03FD	Reserved	—	—	—	—	—	—	—	—
0x(00)00_03FE	Storage of FTRIM	0	0	0	0	0	0	0	FTRIM
0x(00)00_03FF	Storage of ICSTRM	TRIM							
0x(00)00_0400– 0x(00)00_0407		8-Byte Backdoor Comparison Key							
0x(00)00_0408– 0x(00)00_040C	Reserved	—	—	—	—	—	—	—	—
0x(00)00_040D	NVPROT	FPS							FPOPEN
0x(00)00_040E	Reserved	—	—	—	—	—	—	—	—
0x(00)00_040F	NVOPT	KEYEN		0	0	0	0	SEC	

The factory trim values are stored in the flash information row (IFR)[1] and are automatically loaded into the ICSTRM and ICSSC registers after any reset. The oscillator trim values stored in TRIM and FTRIM can be reprogrammed by third party programmers and must be copied into the corresponding ICS registers (ICSTRM and ICSSC) by user code to override the factory trim.

NOTE

When the MCU is in active BDM, the trim value in the IFR is not loaded. Instead, the ICSTRM register resets to 0x80 and ICSSC[FTRIM] resets to zero.

Provided the key enable (KEYEN) bit is set, the 8-byte comparison key can be used to temporarily disengage memory security. This key mechanism can be accessed only through user code running in secure memory (A security key cannot be entered directly through background debug commands). This security key can be disabled completely by clearing the KEYEN bit. If the security key is disabled, the only way

---

1. **IFR** — Nonvolatile information memory that can only be accessed during production test. During production test, system initialization, configuration, and test information is stored in the IFR. This information cannot be read or modified in normal user or background debug modes.

to disengage security is by mass-erasing the flash (normally through the background debug interface) and verifying the flash is blank.

4.2.2 ColdFire Rapid GPIO Memory Map

The rapid GPIO module is mapped into a 16-byte area starting at location 0x(00)C0_0000. Its memory map is shown below in Table 4-6.

Table 4-6. V1 ColdFire Rapid GPIO Memory Map

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x(00)C0_0000	RGPIO_DIR	15	14	13	12	11	10	9	8
0x(00)C0_0001		7	6	5	4	3	2	1	0
0x(00)C0_0002	RGPIO_DATA	15	14	13	12	11	10	9	8
0x(00)C0_0003		7	6	5	4	3	2	1	0
0x(00)C0_0004	RGPIO_ENB	15	14	13	12	11	10	9	8
0x(00)C0_0005		7	6	5	4	3	2	1	0
0x(00)C0_0006	RGPIO_CLR	15	14	13	12	11	10	9	8
0x(00)C0_0007		7	6	5	4	3	2	1	0
0x(00)C0_0008	Reserved	—	—	—	—	—	—	—	—
0x(00)C0_0009		—	—	—	—	—	—	—	—
0x(00)C0_000A	RGPIO_SET	15	14	13	12	11	10	9	8
0x(00)C0_000B		7	6	5	4	3	2	1	0
0x(00)C0_000C	Reserved	—	—	—	—	—	—	—	—
0x(00)C0_000D		—	—	—	—	—	—	—	—
0x(00)C0_000E	RGPIO_TOG	15	14	13	12	11	10	9	8
0x(00)C0_000F		7	6	5	4	3	2	1	0

4.2.3 ColdFire Interrupt Controller Memory Map

The V1 ColdFire interrupt controller (CF1_INTC) register map is sparsely-populated, but retains compatibility with earlier ColdFire interrupt controller definitions. The CF1_INTC occupies the upper 64 bytes of the device memory map and all memory locations are accessed as 8-bit (byte) operands.

Table 4-7. V1 ColdFire Interrupt Controller Memory Map

Address	Register Name	msb			Bit Number				lsb
0x(FF)FF_FFC0–0x(FF)FF_FFD2	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_FFD3	INTC_FRC	0	LVL1	LVL2	LVL3	LVL4	LVL5	LVL6	LVL7
0x(FF)FF_FFD4–0x(FF)FF_FFD7	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_FFD8	INTC_PL6P7	0	0	0	REQN				
0x(FF)FF_FFD9	INTC_PL6P6	0	0	0	REQN				
0x(FF)FF_FFDA	Reserved	—	—	—	—	—	—	—	—
0x(FF)FF_FFDB	INTC_WCR	ENB	0	0	0	0	MASK		

Table 4-7. V1 ColdFire Interrupt Controller Memory Map (continued)

Address	Register Name	msb			Bit Number				lsb
0x(FF)FF_FFDC– 0x(FF)FF_FFDD	**Reserved**	—	—	—	—	—	—	—	—
0x(FF)FF_FFDE	**INTC_SFRC**	0	0	SET					
0x(FF)FF_FFDF	**INTC_CFRC**	0	0	CLR					
0x(FF)FF_FFE0	**INTC_SWIACK**	0	VECN						
0x(FF)FF_FFE1– 0x(FF)FF_FFE3	**Reserved**	—	—	—	—	—	—	—	—
0x(FF)FF_FFE4	**INTC_LVL1IACK**	0	VECN						
0x(FF)FF_FFE5– 0x(FF)FF_FFE7	**Reserved**	—	—	—	—	—	—	—	—
0x(FF)FF_FFE8	**INTC_LVL2IACK**	0	VECN						
0x(FF)FF_FFE9– 0x(FF)FF_FFEB	**Reserved**	—	—	—	—	—	—	—	—
0x(FF)FF_FFEC	**INTC_LVL3IACK**	0	VECN						
0x(FF)FF_FFED– 0x(FF)FF_FFEF	**Reserved**	—	—	—	—	—	—	—	—
0x(FF)FF_FFF0	**INTC_LVL4IACK**	0	VECN						
0x(FF)FF_FFF1– 0x(FF)FF_FFF3	**Reserved**	—	—	—	—	—	—	—	—
0x(FF)FF_FFF4	**INTC_LVL5IACK**	0	VECN						
0x(FF)FF_FFF5– 0x(FF)FF_FFF7	**Reserved**	—	—	—	—	—	—	—	—
0x(FF)FF_FFF8	**INTC_LVL6IACK**	0	VECN						
0x(FF)FF_FFF9– 0x(FF)FF_FFFB	**Reserved**	—	—	—	—	—	—	—	—
0x(FF)FF_FFFC	**INTC_LVL7IACK**	0	VECN						
0x(FF)FF_FFFD– 0x(FF)FF_FFFF	**Reserved**	—	—	—	—	—	—	—	—

4.3   RAM

The MCF51QE128/64/32 includes up to 8 Kbytes of static RAM. RAM is most efficiently accessed using the A5-relative addressing mode (address register indirect with displacement mode). Any single bit in this area can be accessed with the bit manipulation instructions (BCLR, BSET,etc.).

At power-on, the contents of RAM are uninitialized. RAM data is unaffected by any reset provided that the supply voltage does not drop below the minimum value for RAM retention (V_{RAM}).

4.4   Flash

The flash memory is intended primarily for program storage and read-only data. In-circuit programming allows the operating program to be loaded into the flash memory after final assembly of the application product. It is possible to program the entire array through the single-wire background debug interface. Because no special voltages are needed for flash erase and programming operations, in-application programming is also possible through other software-controlled communication paths.

Flash memory is ideal for single-supply applications allowing for field reprogramming without requiring external high voltage sources for program or erase operations. The flash module includes a memory controller that executes commands to modify flash memory contents.

Array read access time is one bus cycle for bytes, aligned words, and aligned longwords. Multiple accesses are needed for misaligned words and longword operands. For flash memory, an erased bit reads 1 and a programmed bit reads 0. It is not possible to read from a flash block while any command is executing on that specific flash block.

CAUTION

A flash block address must be in the erased state before being programmed. Cumulative programming of bits within a flash block address is not allowed except for status field updates required in EEPROM emulation applications.

Flash memory on MCF51QE128/64/32 must be programmed 32-bits at a time when the low-voltage detect flag (LVDF) in the system power management status and control 1 register (SPMSC1) is clear. If SPMSC1[LVDF] is set, the programming sequence must be modified such that odd and even bytes are written separately. The MCF51QE128/64/32 flash memory is organized as two 16-bit wide blocks interleaved to yield a 32-bit data path. When programming flash when LVDF is set, alternate bytes must be set to 0xFF as shown in Table 4-8. Failure to adhere to these guidelines may result in a partially programmed flash array.

Table 4-8. Lov-Voltage Programming Sequence Example

Addresses	Desired Value	Values Programmed
0x00 – 0x03 0x00 – 0x03	0x5555_AAAA	0x55FF_AAFF 0xFF55_FFAA
0x04 – 0x07 0x04 – 0x07	0xCCCC_CCCC	0xCCFF_CCFF 0xFFCC_FFCC
0x08 – 0x0B 0x08 – 0x0B	0x1234_5678	0x12FF_56FF 0xFF34_FF78
0x0C – 0x0F 0x0C – 0x0F	0x9ABC_DEF0	0x9AFF_DEFF 0xFFBC_FFF0

4.4.1   Features

Features of the flash memory include:

- Flash size
  — MCF51QE128: 131,072 bytes (128 sectors of 1024 bytes each)
  — MCF51QE64: 65,536 bytes (64 sectors of 1024 bytes each)
  — MCF51QE32: 32,768 bytes (32 sectors of 1024 bytes each)
- Automated program and erase algorithm
- Fast program and sector erase operation
- Burst program command for faster flash array program times
- Single power supply program and erase

MCF51QE128 MCU Series Reference Manual, Rev. 3

- Command interface for fast program and erase operation
- Up to 100,000 program/erase cycles at typical voltage and temperature
- Flexible block protection (on any 2-Kbyte memory boundary)
- Security feature to prevent unauthorized access to on-chip memory and resources
- Auto power-down for low-frequency read accesses

4.4.2 Register Descriptions

The flash module contains a set of 16 control and status registers located between 0x(00)00_0000 and 0x(00)00_000F. Detailed descriptions of each register bit are provided in the following sections.

4.4.2.1 Flash Clock Divider Register (FCDIV)

The FCDIV register controls the length of timed events in program and erase algorithms executed by the flash memory controller. All bits in the FCDIV register are readable and writable with restrictions as determined by the value of FDIVLD when writing to the FCDIV register.

	7	6	5	4	3	2	1	0
R	FDIVLD	PRDIV8			FDIV			
W								
Reset	0	0	0	0	0	0	0	0
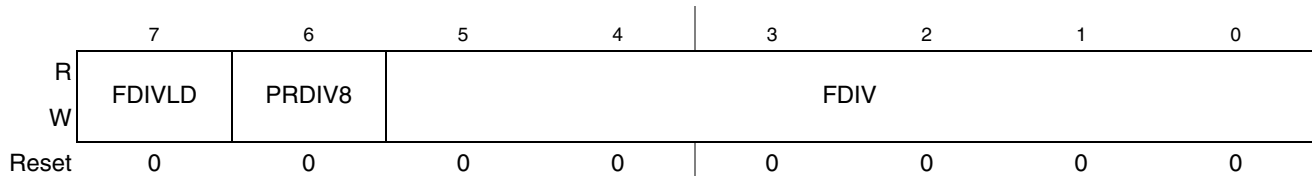
Figure 4-3. Flash Clock Divider Register (FCDIV)

Table 4-9. FCDIV Field Descriptions

Field	Description
7 FDIVLD	**Clock Divider Load Control**. When writing to the FCDIV register for the first time after a reset, the value of the FDIVLD bit written controls the future ability to write to the FCDIV register: 0 Writing a 0 to FDIVLD locks the FCDIV register contents; all future writes to FCDIV are ignored. 1 Writing a 1 to FDIVLD keeps the FCDIV register writable; next write to FCDIV is allowed. When reading the FCDIV register, the value of the FDIVLD bit read indicates the following: 0 FCDIV register has not been written to since the last reset. 1 FCDIV register has been written to since the last reset.
6 PRDIV8	**Enable Prescalar by 8.** 0 The bus clock is directly fed into the clock divider. 1 The bus clock is divided by 8 before feeding into the clock divider.
5–0 FDIV	**Clock Divider Bits**. The combination of PRDIV8 and FDIV[5:0] must divide the bus clock down to a frequency of 150 kHz–200 kHz. The minimum divide ratio is 2 (PRDIV8=0, FDIV=0x01) and the maximum divide ratio is 512 (PRDIV8=1, FDIV=0x3F). Refer to Section 4.5.1.1, "Writing the FCDIV Register" for more information.

4.4.2.2 Flash Options Register (FOPT and NVOPT)

The FOPT register holds all bits associated with the security of the MCU and flash module. All bits in the FOPT register are readable but are not writable.

The FOPT register is loaded from the flash configuration field (see Section 4.2.1) during the reset sequence, indicated by F in Figure 4-4.

The security feature in the flash module is described in Section 4.5.5, "Security".

	7	6	5	4	3	2	1	0
R	KEYEN		0	0	0	0	SEC	
W								
Reset	F	F	0	0	0	0	F	F
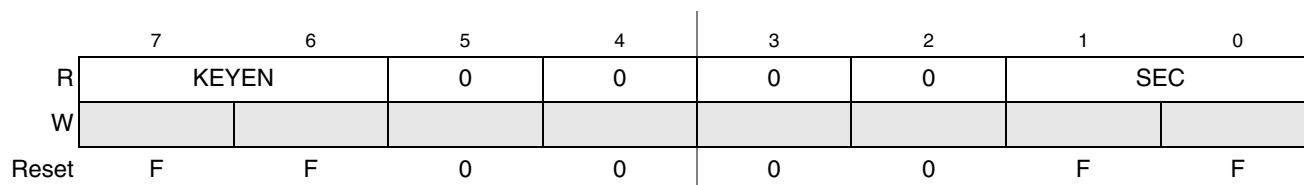
Figure 4-4. Flash Options Register (FOPT)

Table 4-10. FOPT Field Descriptions

Field	Description
7–6 KEYEN	**Backdoor Key Security Enable Bits**. The KEYEN[1:0] bits define the enabling of backdoor key access to the flash module. 00 Disabled 01 Disabled (Preferred KEYEN state to disable Backdoor Key Access) 10 Enabled 11 Disabled
5–2	Reserved, should be cleared.
1–0 SEC	**Flash Security Bits**. The SEC[1:0] bits define the security state of the MCU. If the flash module is unsecured using backdoor key access, the SEC[1:0] bits are forced to the unsecured state. 00 Unsecured 01 Unsecured 10 Secured 11 Unsecured

4.4.2.3 Flash Configuration Register (FCNFG)

The FCNFG register gates the security backdoor writes.

KEYACC is readable and writable while all remaining bits read 0 and are not writable. KEYACC is only writable if KEYEN is set to the enabled state (see Section 4.4.2.2, "Flash Options Register (FOPT and NVOPT)").

NOTE

Flash array reads are allowed while KEYACC is set.

	7	6	5	4	3	2	1	0
R	0	0	KEYACC	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0
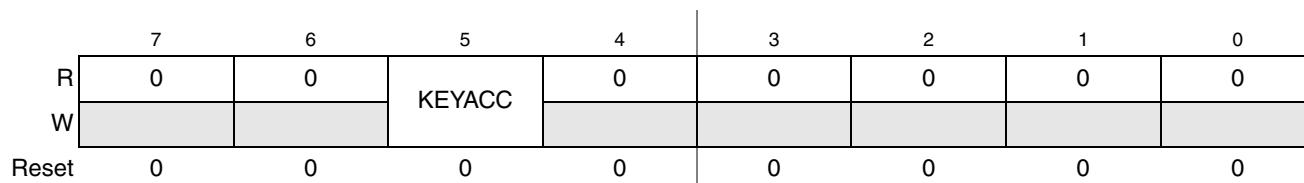
Figure 4-5. Flash Configuration Register (FCNFG)

Table 4-11. FCNFG Field Descriptions

Field	Description
7–6	Reserved, should be cleared.
5 KEYACC	Enable Security Key Writing 0 Writes to the flash block are interpreted as the start of a command write sequence. 1 Writes to the flash block are interpreted as keys to open the backdoor.
4–0	Reserved, should be cleared.

4.4.2.4　Flash Protection Register (FPROT and NVPROT)

The FPROT register defines which flash sectors are protected against program or erase operations. FPROT bits are readable and writable as long as the size of the protected flash memory is being increased. Any write to FPROT that attempts to decrease the size of the protected flash memory is ignored.

During the reset sequence, the FPROT register is loaded from the flash protection byte in the flash configuration field (see Section 4.2.1), indicated by F in Table 4-6. To change the flash protection loaded during the reset sequence, the flash sector containing the flash configuration field must be unprotected. Then, the flash protection byte must be reprogrammed.

Trying to alter data in any protected area in the flash memory results in a protection violation error and FSTAT[FPVIOL] is set. The mass erase of the flash array is not possible if any of the flash sectors contained in the flash array are protected.

	7	6	5	4	3	2	1	0
R W				FPS				FPOPEN
Reset	F	F	F	F	F	F	F	F
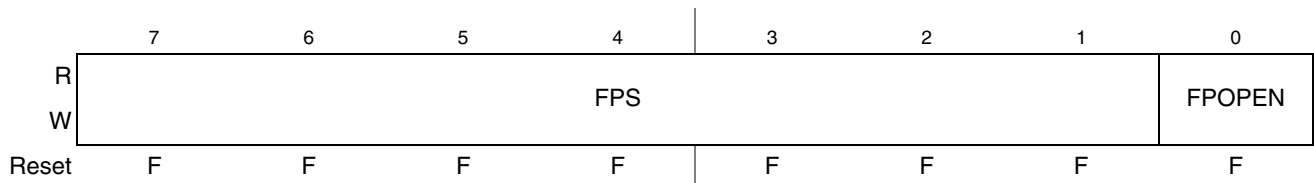
Figure 4-6. Flash Protection Register (FPROT)

Table 4-12. FPROT Field Descriptions

Field	Description
7–1 FPS	**Flash Protection Size**. With FPOPEN set, the FPS bits determine the size of the protected flash address range as shown in Table 4-13.
0 FPOPEN	Flash Protection Open 0 Flash array fully protected. 1 Flash array protected address range determined by FPS bits.

Table 4-13. Flash Protection Address Range

FPS	FPOPEN	Protected Address Range Relative to Flash Array Base	Protected Size
—	0	0x0_0000–0x1_FFFF	128 Kbytes

Table 4-13. Flash Protection Address Range (continued)

FPS	FPOPEN	Protected Address Range Relative to Flash Array Base	Protected Size
0x00–0x3F	1	0x0_0000–0x1_FFFF	128 Kbytes
0x40		0x0_0000–0x1_F7FF	126 Kbytes
0x41		0x0_0000–0x1_EFFF	124 Kbytes
0x42		0x0_0000–0x1_E7FF	122 Kbytes
0x43		0x0_0000–0x1_DFFF	120 Kbytes
0x44		0x0_0000–0x1_D7FF	118 Kbytes
0x45		0x0_0000–0x1_CFFF	116 Kbytes
0x46		0x0_0000-0x1_C7FF	114 Kbytes
0x47	1	0x0_0000–0x1_BFFF	112 Kbytes
...		...	...
0x5B		0x0_0000–0x1_1FFF	72 Kbytes
0x5C		0x0_0000–0x1_17FF	70 Kbytes
0x5D		0x0_0000–0x1_0FFF	68 Kbytes
0x5E		0x0_0000–0x1_07FF	66 Kbytes
0x5F		0x0_0000–0x0_FFFF	64 Kbytes
0x60		0x0_0000–0x0_F7FF	62 Kbytes
0x61		0x0_0000–0x0_EFFF	60 Kbytes
0x62		0x0_0000–0x0_E7FF	58 Kbytes
0x63		0x0_0000–0x0_DFFF	56 Kbytes
...		...	...
0x77		0x0_0000–0x0_3FFF	16 Kbytes
0x78		0x0_0000–0x0_37FF	14 Kbytes
0x79		0x0_0000–0x0_2FFF	12 Kbytes
0x7A		0x0_0000–0x0_27FF	10 Kbytes
0x7B		0x0_0000–0x0_1FFF	8 Kbytes
0x7C		0x0_0000–0x0_17FF	6 Kbytes
0x7D		0x0_0000–0x0_0FFF	4 Kbytes
0x7E		0x0_0000–0x0_07FF	2 Kbytes
0x7F		No Protection	0 Kbytes

MCF51QE128 MCU Series Reference Manual, Rev. 3

Get the latest version from freescale.com

4.4.2.5 Flash Status Register (FSTAT)

The FSTAT register defines the operational status of the flash module. FCBEF, FPVIOL and FACCERR are readable and writable. FBLANK is readable and not writable. The remaining bits read 0 and are not writable.

	7	6	5	4	3	2	1	0
R	FCBEF	FCCF	FPVIOL	FACCERR	0	FBLANK	0	0
W	w1c		w1c	w1c				
Reset	1	1	0	0	0	0	0	0

Figure 4-7. Flash Status Register (FSTAT)

Table 4-14. FSTAT Field Descriptions

Field	Description
7 FCBEF	**Command Buffer Empty Flag.** The FCBEF flag indicates that the command buffer is empty so that a new command write sequence can be started when performing burst programming. Writing a 0 to the FCBEF flag has no effect on FCBEF. Writing a 0 to FCBEF after writing an aligned address to the flash array memory, but before FCBEF is cleared, aborts a command write sequence and causes the FACCERR flag to be set. Writing a 0 to FCBEF outside of a command write sequence does not set the FACCERR flag. Writing a 1 to this bit clears it. 0 Command buffers are full. 1 Command buffers are ready to accept a new command.
6 FCCF	**Command Complete Flag**. The FCCF flag indicates that there are no more commands pending. The FCCF flag is cleared when FCBEF is cleared and sets automatically upon completion of all active and pending commands. The FCCF flag does not set when an active program command completes and a pending burst program command is fetched from the command buffer. Writing to the FCCF flag has no effect on FCCF. 0 Command in progress. 1 All commands are completed.
5 FPVIOL	**Protection Violation Flag**. The FPVIOL flag indicates an attempt was made to program or erase an address in a protected area of the flash memory during a command write sequence. Writing a 0 to the FPVIOL flag has no effect on FPVIOL. Writing a 1 to this bit clears it. While FPVIOL is set, it is not possible to launch a command or start a command write sequence. 0 No protection violation detected. 1 Protection violation has occurred.
4 FACCERR	**Access Error Flag**. The FACCERR flag indicates an illegal access has occurred to the flash memory caused by either a violation of the command write sequence (see Section 4.5.1.2, "Command Write Sequence"), issuing an illegal flash command (see Section 4.4.2.6, "Flash Command Register (FCMD)"), or the execution of a CPU STOP instruction while a command is executing (FCCF = 0). Writing a 0 to the FACCERR flag has no effect on FACCERR. Writing a 1 to this bit clears it. While FACCERR is set, it is not possible to launch a command or start a command write sequence. 0 No access error detected. 1 Access error has occurred.
3	Reserved, should be cleared.

Table 4-14. FSTAT Field Descriptions (continued)

Field	Description
2 FBLANK	**Flag Indicating the Erase Verify Operation Status**. When the FCCF flag is set after completion of an erase verify command, the FBLANK flag indicates the result of the erase verify operation. The FBLANK flag is cleared by the flash module when FCBEF is cleared as part of a new valid command write sequence. Writing to the FBLANK flag has no effect on FBLANK. 0 Flash block verified as not erased. 1 Flash block verified as erased.
1–0	Reserved, should be cleared.

4.4.2.6   Flash Command Register (FCMD)

The FCMD register is the flash command register. All FCMD bits are readable and writable during a command write sequence while bit 7 reads 0 and is not writable.

	7	6	5	4	3	2	1	0
R	0				FCMD			
W								
Reset	0	0	0	0	0	0	0	0
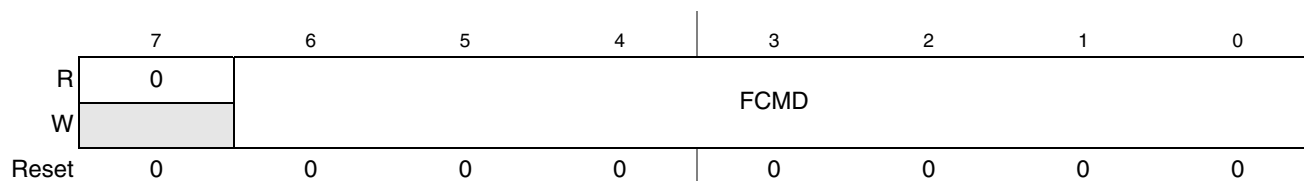
Figure 4-8. Flash Command Register (FCMD)

Table 4-15. FCMD Field Descriptions

Field	Description
7	Reserved, should be cleared.
6–0 FCMD	**Flash Command.** Valid flash commands are shown below. Writing any command other than those listed sets the FACCERR flag in the FSTAT register. 0x05 Erase Verify 0x20 Program 0x25 Burst Program 0x40 Sector Erase 0x41 Mass Erase

4.5   Function Description

4.5.1   Flash Command Operations

Flash command operations execute program, erase, and erase verify algorithms described in this section. The program and erase algorithms are controlled by the flash memory controller whose time base, FCLK, is derived from the bus clock via a programmable divider.

The next sections describe:

1. How to write the FCDIV register to set FCLK
2. Command write sequences to program, erase, and erase verify operations on the flash memory
3. Valid flash commands

    4.   Effects resulting from illegal flash command write sequences or aborting flash operations

4.5.1.1   Writing the FCDIV Register

Prior to issuing any flash command after a reset, write the FCDIV register to divide the bus clock down to 150–200 kHz. The FCDIV[PRDIV8, FDIV] bits must be set as described in Figure 4-9.

For example, if the bus clock frequency is 25 MHz, FCDIV[FDIV] should be set to 0x0F (001111) and the FCDIV[PRDIV8] bit set to 1. The resulting FCLK frequency is then 195 kHz. In this case, the flash program and erase algorithm timings are increased over the optimum target by:

$$(200 - 195) \div 200 = 3\% \qquad\qquad\qquad \textit{Eqn. 4-1}$$

CAUTION

    Program and erase command execution time increase proportionally with the period of FCLK. Programming or erasing the flash memory with FCLK less than 150 kHz should be avoided. Setting FCDIV to a value such that FCLK is less than 150 kHz can destroy the flash memory due to overstress. Setting FCDIV to a value where FCLK is greater than 200 kHz can result in incomplete programming or erasure of the flash memory cells.

If the FCDIV register is written, the FDIVLD bit is automatically set. If the FDIVLD bit is 0, the FCDIV register has not been written since the last reset. If the FCDIV register has not been written to, the flash command loaded during a command write sequence does not execute and FSTAT[FACCERR] is set.
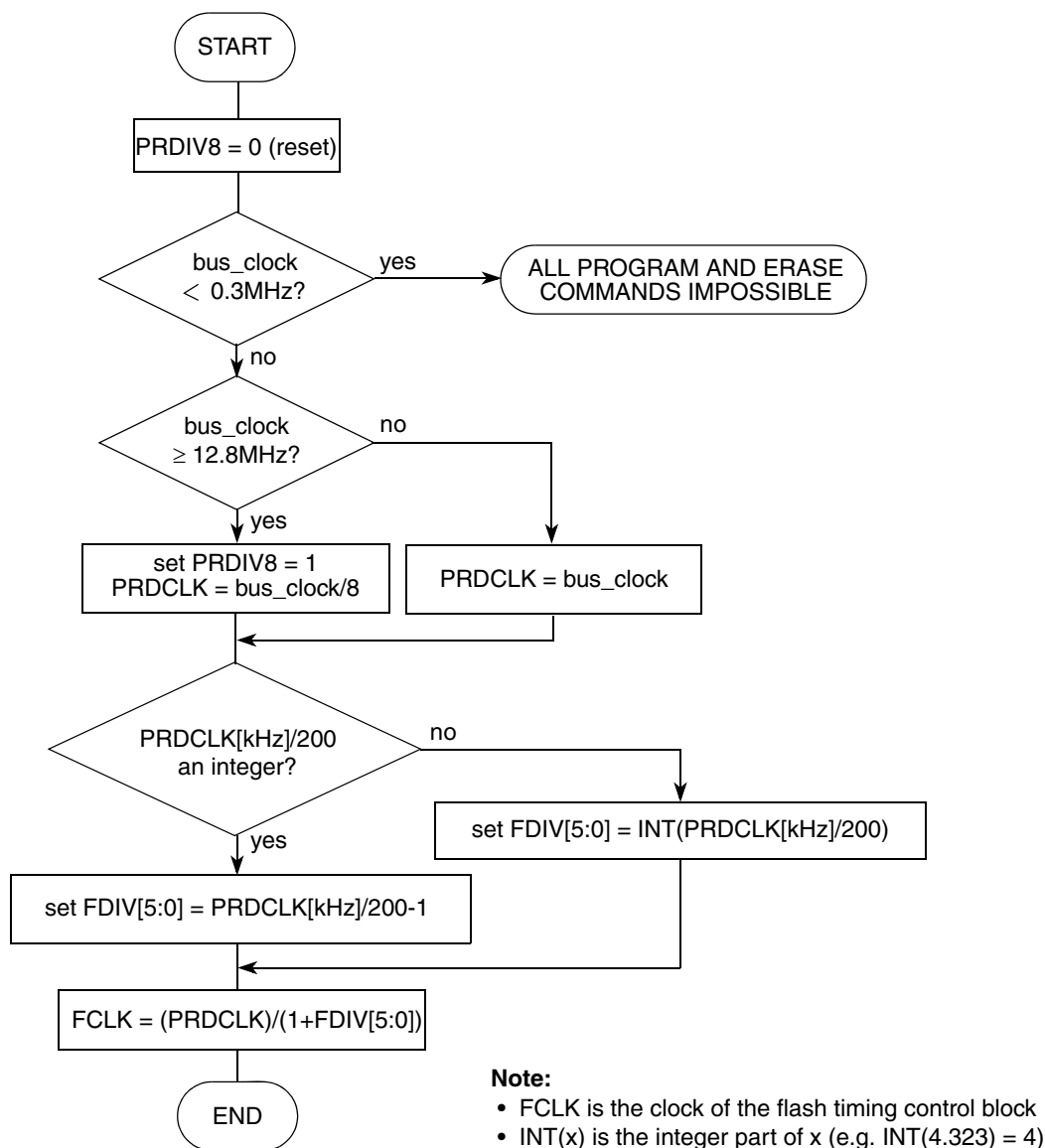
Figure 4-9. Determination Procedure for PRDIV8 and FDIV Bits

4.5.1.2   Command Write Sequence

The flash command controller supervises the command write sequence to execute program, erase, and erase verify algorithms.

Before starting a command write sequence, the FACCERR and FPVIOL flags in the FSTAT register must be clear and the FCBEF flag must be set (see Section 4.4.2.5).

A command write sequence consists of three steps that must be strictly adhered to with writes to the flash module not permitted between the steps. However, flash register and array reads are allowed during a command write sequence. The basic command write sequence is as follows:

1. Write to a valid address in the flash array memory.

2. Write a valid command to the FCMD register.
3. Clear the FCBEF flag in the FSTAT register by writing a 1 to FCBEF to launch the command.

After a command is launched, the completion of the command operation is indicated by the setting of FSTAT[FCCF]. The FCCF flag sets upon completion of all active and buffered burst program commands.

4.5.2   Flash Commands

Table 4-16 summarizes the valid flash commands along with the effects of the commands on the flash block.

Table 4-16. Flash Command Description

FCMD	NVM Command	Function on Flash Memory
0x05	Erase Verify	Verify all memory bytes in the flash array memory are erased. If the flash array memory is erased, FSTAT[FBLANK] sets upon command completion.
0x20	Program	Program an address in the flash array.
0x25	Burst Program	Program an address in the flash array with the internal address incrementing after the program operation.
0x40	Sector Erase	Erase all memory bytes in a sector of the flash array.
0x41	Mass Erase	Erase all memory bytes in the flash array. A mass erase of the full flash array is only possible when no protection is enabled prior to launching the command.

CAUTION

A flash block address must be in the erased state before being programmed. Cumulative programming of bits within a flash block address is not allowed except for status field updates required in EEPROM emulation applications.

4.5.2.1   Erase Verify Command

The erase verify operation verifies that the entire flash array memory is erased.

An example flow to execute the erase verify operation is shown in Figure 4-10. The erase verify command write sequence is as follows:

1. Write to an aligned flash block address to start the command write sequence for the erase verify command. The address and data written are ignored.
2. Write the erase verify command, 0x05, to the FCMD register.
3. Clear the FCBEF flag in the FSTAT register by writing a 1 to FCBEF to launch the erase verify command.

After launching the erase verify command, FSTAT[FCCF] sets after the operation has completed. The number of bus cycles required to execute the erase verify operation is equal to the number of addresses in the flash array memory plus several bus cycles as measured from the time the FCBEF flag is cleared until the FCCF flag is set. Upon completion of the erase verify operation, FSTAT[FBLANK] is set if all addresses in the flash array memory are verified to be erased. If any address in the flash array memory is not erased, the erase verify operation terminates and FSTAT[FBLANK] remains cleared.
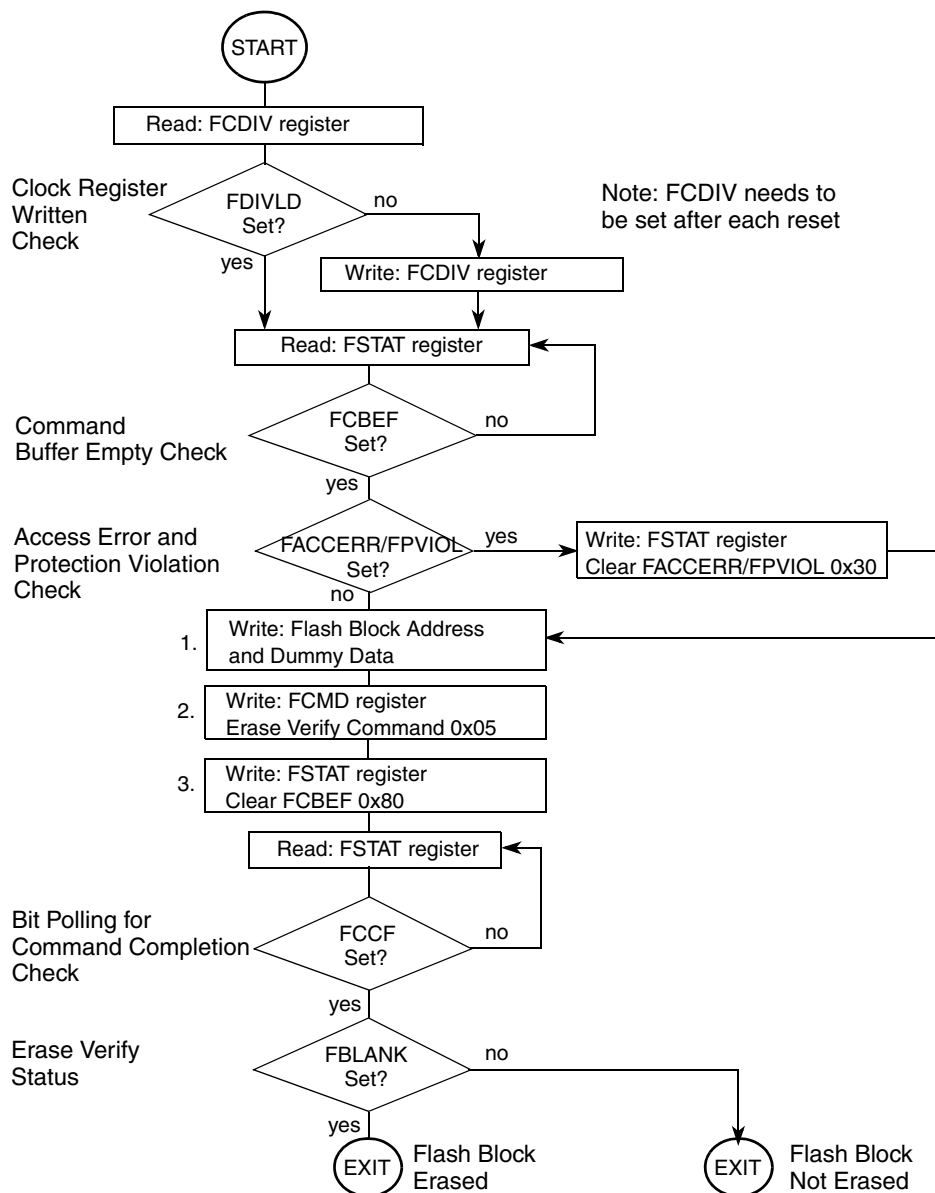
Figure 4-10. Example Erase Verify Command Flow

4.5.2.2 Program Command

The program operation programs a previously erased address in the flash memory using an embedded algorithm. An example flow to execute the program operation is shown in Figure 4-11. The program command write sequence is as follows:

1. Write to an aligned flash block address to start the command write sequence for the program command. The data written is programmed to the address written.

2. Write the program command, 0x20, to the FCMD register.

3. Clear the FCBEF flag in the FSTAT register by writing a 1 to FCBEF to launch the program command.

MCF51QE128 MCU Series Reference Manual, Rev. 3

Get the latest version from freescale.com

If an address to be programmed is in a protected area of the flash block, FSTAT[FPVIOL] sets and the program command does not launch. After the program command has successfully launched and the program operation has completed, FSTAT[FCCF] is set.
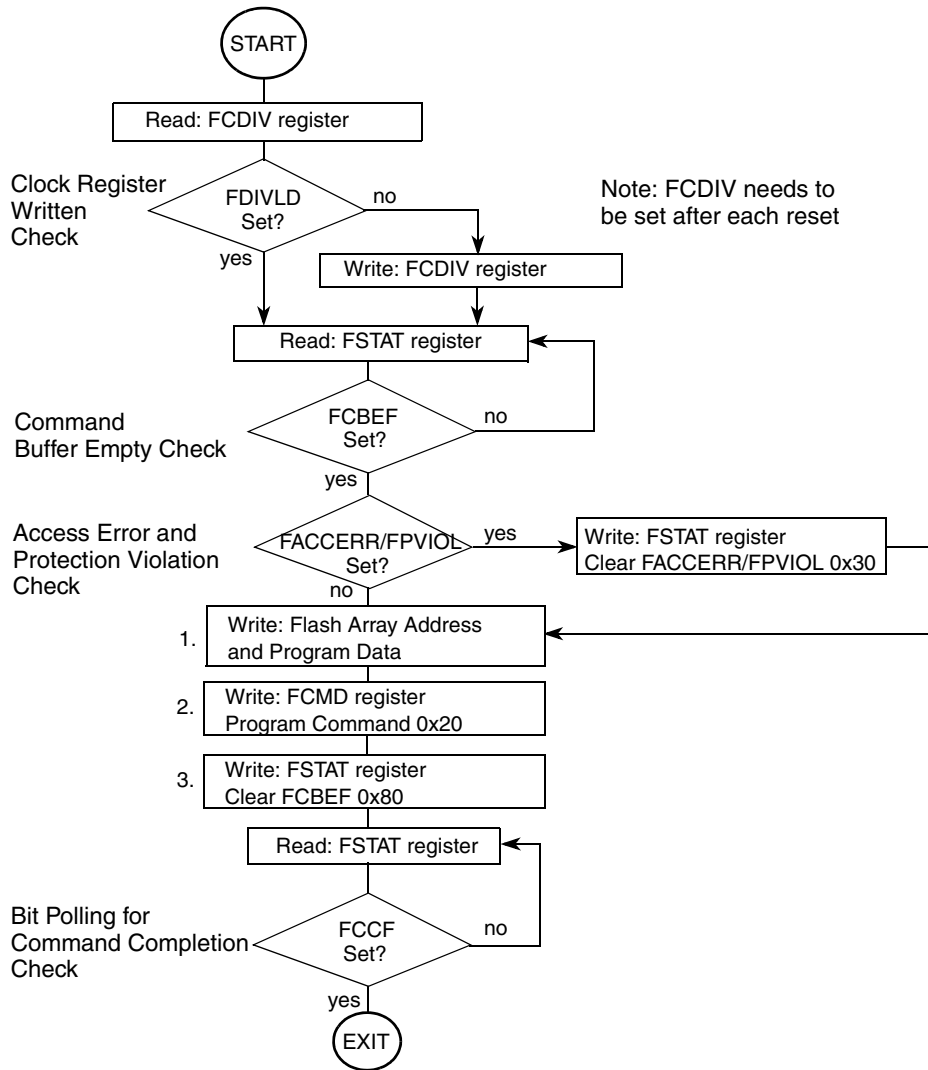
Figure 4-11. Example Program Command Flow

4.5.2.3   Burst Program Command

The burst program operation programs previously erased data in the flash memory using an embedded algorithm.

While burst programming, two internal data registers operate as a buffer and a register (2-stage FIFO) so that a second burst programming command along with the necessary data can be stored to the buffers while the first burst programming command remains in progress. This pipelined operation allows a time optimization when programming more than one consecutive address on a specific row in the flash array as the high voltage generation can be kept active in between two programming commands.

An example flow to execute the burst program operation is shown in Figure 4-12. The burst program command write sequence is as follows:

1.   Write to an aligned flash block address to start the command write sequence for the burst program command. The data written is programmed to the address written.
2.   Write the program burst command, 0x25, to the FCMD register.
3.   Clear the FCBEF flag in the FSTAT register by writing a 1 to FCBEF to launch the program burst command.
4.   After the FCBEF flag in the FSTAT register returns to a 1, repeat steps 1 through 3. The address written is ignored but is incremented internally.

The burst program procedure can be used to program the entire flash memory even while crossing row boundaries within the flash array. If data to be burst programmed falls within a protected area of the flash array, FSTAT[FPVIOL] is set and the burst program command does not launch. After the burst program command has successfully launched and the burst program operation has completed, FSTAT[FCCF] is set unless a new burst program command write sequence has been buffered. By executing a new burst program command write sequence on sequential addresses after the FCBEF flag in the FSTAT register has been set, a greater than 50% faster programming time for the entire flash array can be effectively achieved when compared to using the basic program command.
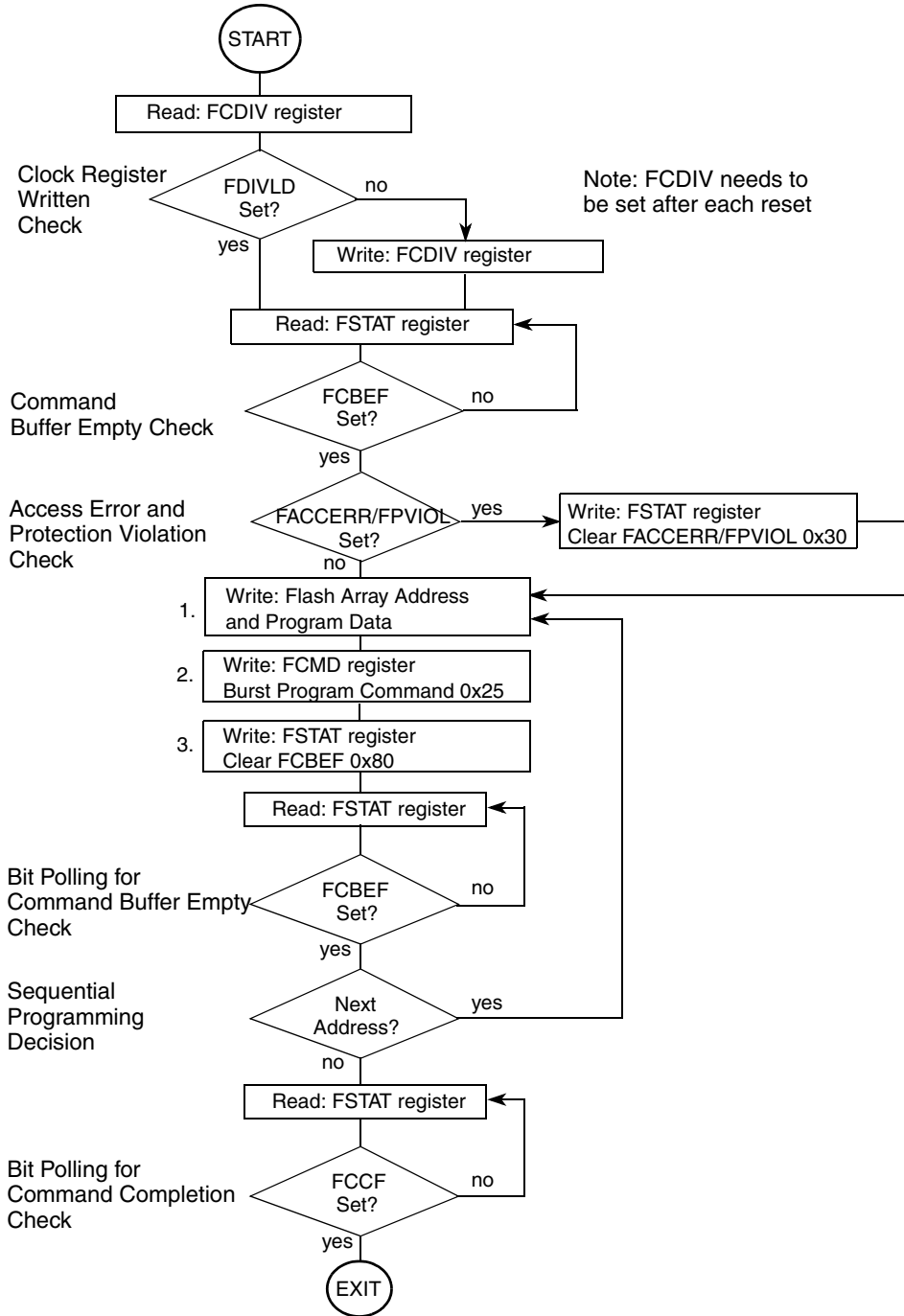
Figure 4-12. Example Burst Program Command Flow

4.5.2.4　Sector Erase Command

The sector erase operation erases all addresses in a 1 Kbyte sector of flash memory using an embedded algorithm.

An example flow to execute the sector erase operation is shown in Figure 4-13. The sector erase command write sequence is as follows:

1. Write to an aligned flash block address to start the command write sequence for the sector erase command. The flash address written determines the sector to be erased while the data written is ignored.

2. Write the sector erase command, 0x40, to the FCMD register.

3. Clear the FCBEF flag in the FSTAT register by writing a 1 to FCBEF to launch the sector erase command.

If a flash sector to be erased is in a protected area of the flash block, FSTAT[FPVIOL] is set and the sector erase command does not launch. After the sector erase command has successfully launched and the sector erase operation has completed, FSTAT[FCCF] is set.
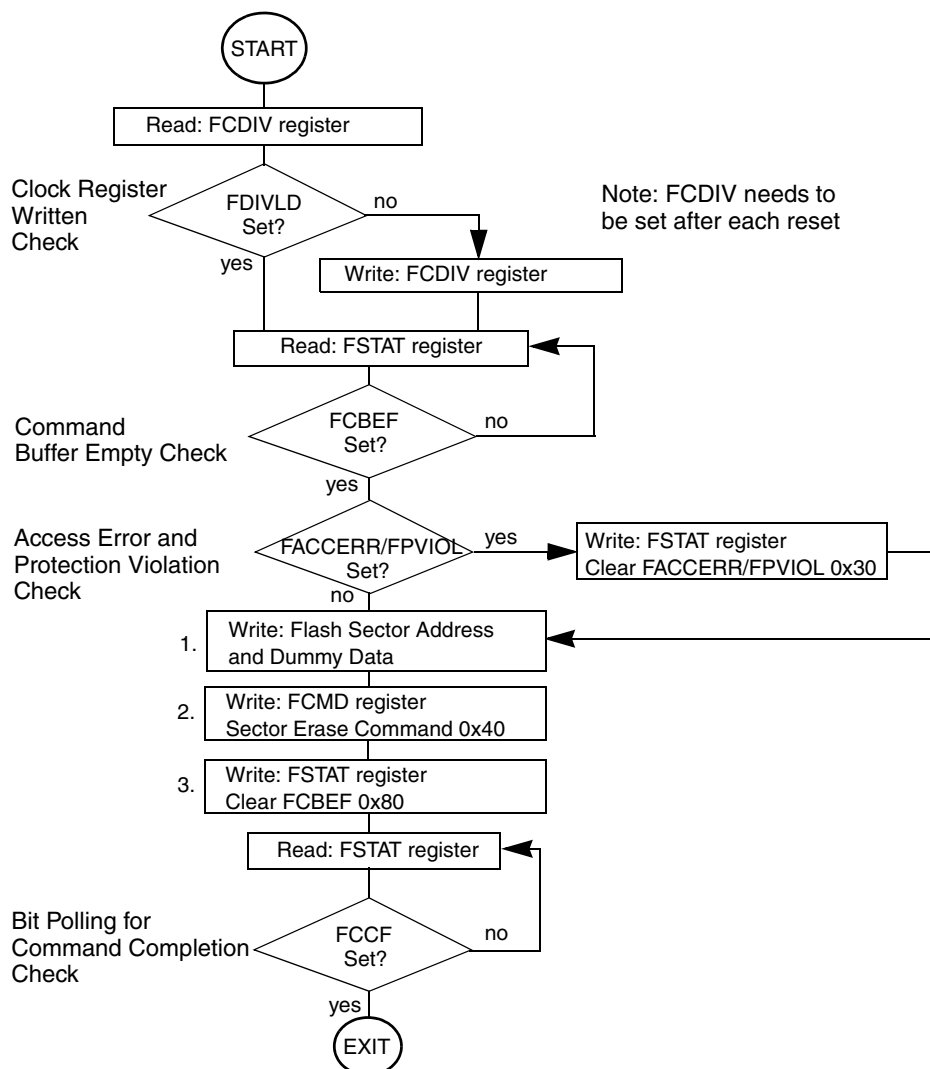
Figure 4-13. Example Sector Erase Command Flow

4.5.2.5   Mass Erase Command

The mass erase operation erases the entire flash array memory using an embedded algorithm. An example flow to execute the mass erase operation is shown in Figure 4-14. The mass erase command write sequence is as follows:

1. Write to an aligned flash block address to start the command write sequence for the mass erase command. The address and data written is ignored.
2. Write the mass erase command, 0x41, to the FCMD register.
3. Clear the FCBEF flag in the FSTAT register by writing a 1 to FCBEF to launch the mass erase command.

If the flash array memory to be mass erased contains any protected area, FSTAT[FPVIOL] is set and the mass erase command does not launch. After the mass erase command has successfully launched and the mass erase operation has completed, FSTAT[FCCF] is set.
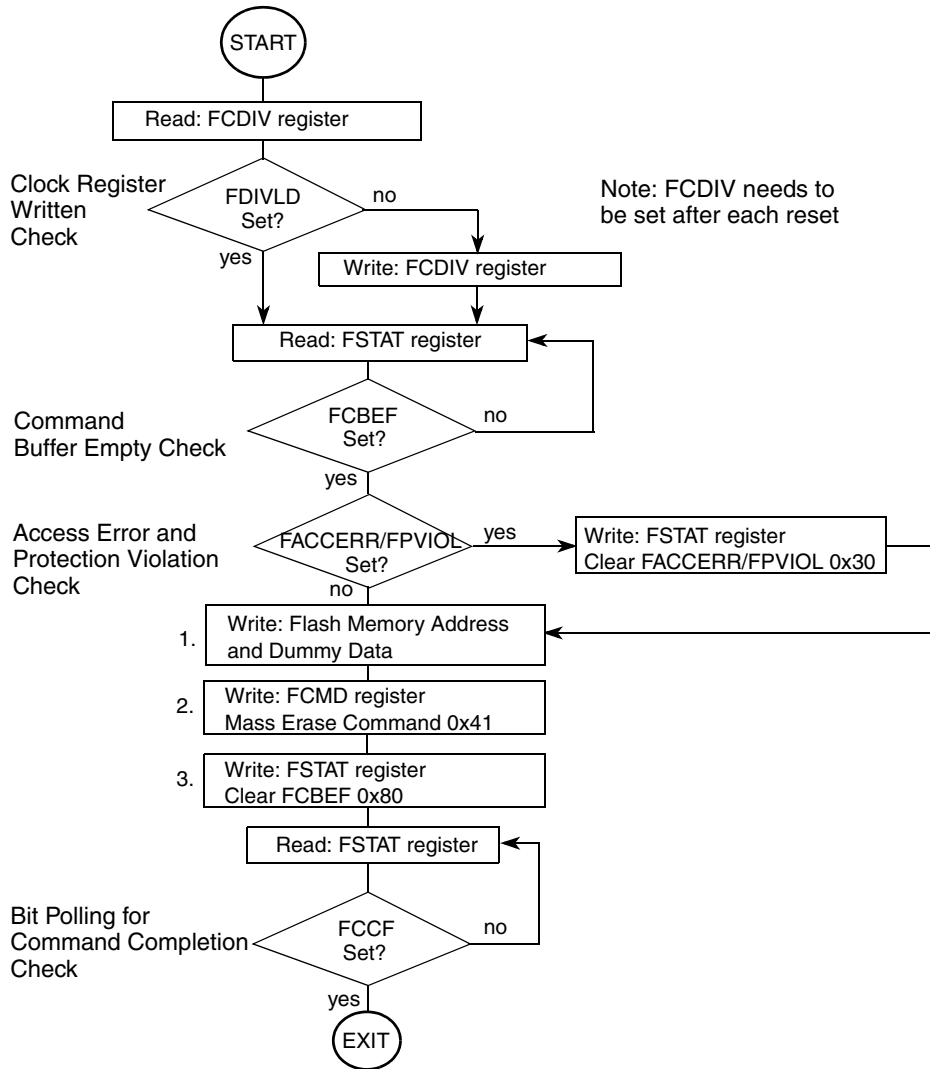
Figure 4-14. Example Mass Erase Command Flow

MCF51QE128 MCU Series Reference Manual, Rev. 3

NOTE

The BDM can also perform a mass erase and verify command. See
Chapter 18, "Version 1 ColdFire Debug (CF1_DEBUG)," for details.

4.5.3   Illegal Flash Operations

4.5.3.1   Flash Access Violations

The FACCERR flag is set during the command write sequence if any of the following illegal steps are
performed, causing the command write sequence to immediately abort:

1. Writing to a flash address before initializing the FCDIV register.
2. Writing a byte, word, or misaligned longword to a valid flash address.
3. Writing to any flash register other than FCMD after writing to a flash address.
4. Writing to a second flash address in the same command write sequence.
5. Writing an invalid command to the FCMD register unless the address written was in a protected
   area of the flash array.
6. Writing a command other than burst program while FCBEF is set and FCCF is clear.
7. When security is enabled, writing a command other than erase verify or mass erase to the FCMD
   register when the write originates from a non-secure memory location or from the background
   debug mode.
8. Writing to a flash address after writing to the FCMD register.
9. Writing to any flash register other than FSTAT (to clear FCBEF) after writing to the FCMD
   register.
10. Writing a 0 to the FCBEF flag in the FSTAT register to abort a command write sequence.

The FACCERR flag is also set if the MCU enters stop mode while any command is active (FCCF=0). The
operation is aborted immediately and, if burst programming, any pending burst program command is
purged (see Section 4.5.4.2, "Stop Modes").

The FACCERR flag does not set if any flash register is read during a valid command write sequence.

If the flash memory is read during execution of an algorithm (FCCF = 0), the read operation returns invalid
data and the FACCERR flag is not set.

If the FACCERR flag is set in the FSTAT register, clear the FACCERR flag before starting another
command write sequence (see Section 4.4.2.5, "Flash Status Register (FSTAT)").

4.5.3.2 Flash Protection Violations

The FPVIOL flag is set after the command is written to the FCMD register if any of the following illegal operations are attempted:

1. Writing the program command if the address written in the command write sequence was in a protected area of the flash array.
2. Writing the sector erase command if the address written in the command write sequence was in a protected area of the flash array.
3. Writing the mass erase command while any flash protection is enabled.
4. Writing an invalid command if the address written in the command write sequence was in a protected area of the flash array.

As a result of any of the above, the command write sequence immediately aborts. If FSTAT[FPVIOL] is set, clear the FPVIOL flag before starting another command write sequence (see Section 4.4.2.5, "Flash Status Register (FSTAT)").

4.5.4 Operating Modes

4.5.4.1 Wait Mode

If a command is active (FCCF = 0) when the MCU enters wait mode, the active command and any buffered command is completed.

4.5.4.2 Stop Modes

If a command is active (FCCF = 0) when the MCU enters any stop mode, the operation is aborted. If the operation is program or erase, the flash array data being programmed or erased may be corrupted and the FCCF and FACCERR flags are set. If active, the high voltage circuitry to the flash array is immediately switched off when entering stop mode. Upon exit from stop mode, the FCBEF flag is set and any buffered command is not launched. The FACCERR flag must be cleared before starting a command write sequence (see Section 4.5.1.2, "Command Write Sequence").

NOTE

As active commands are immediately aborted when the MCU enters stop mode, do not use the STOP instruction during program or erase operations.

Active commands continue when the MCU enters wait mode. Use of the STOP instruction when SOPT1[WAITE] is set is acceptable.

4.5.4.3 Background Debug Mode

In background debug mode, the FPROT register is writable without restrictions. If the MCU is unsecured, all flash commands listed in Table 4-16 can be executed. If the MCU is secured, only a compound mass erase and erase verify command can be executed. See Chapter 18, "Version 1 ColdFire Debug (CF1_DEBUG)," for details.

4.5.5 Security

The flash module provides the necessary security information to the MCU. During each reset sequence, the flash module determines the security state of the MCU as defined in Section 4.2.1, "Flash Module Reserved Memory Locations".

The contents of the flash security byte in the flash configuration field (see Section 4.4.2.3) must be changed directly by programming the flash security byte location when the MCU is unsecured and the sector containing the flash security byte is unprotected. If the flash security byte is left in a secured state, any reset causes the MCU to initialize into a secure operating mode.

4.5.5.1 Unsecuring the MCU using Backdoor Key Access

The MCU may be unsecured by using the backdoor key access feature that requires knowledge of the contents of the backdoor keys (see Section 4.2.1). If the KEYEN[1:0] bits are in the enabled state (see Section 4.4.2.2) and the KEYACC bit is set, a write to a backdoor key address in the flash memory triggers a comparison between the written data and the backdoor key data stored in the flash memory. If all backdoor keys are written to the correct addresses in the correct order and the data matches the backdoor keys stored in the flash memory, the MCU is unsecured. The data must be written to the backdoor keys sequentially. Values 0x0000_0000 and 0xFFFF_FFFF are not permitted as backdoor keys. While the KEYACC bit is set, reads of the flash memory return valid data.

The user code stored in the flash memory must have a method of receiving the backdoor keys from an external stimulus. This external stimulus would typically be through one of the on-chip serial ports.

If the KEYEN[1:0] bits are in the enabled state (see Section 4.4.2.2), the MCU can be unsecured by the backdoor key access sequence described below:

1. Set FCNFG[KEYACC].
2. Execute three NOP instructions to provide time for the backdoor state machine to load the starting address and number of keys required into the flash state machine.
3. Sequentially write the correct longwords to the flash address(es) containing the backdoor keys.
4. Clear the KEYACC bit. Depending on the user code used to write the backdoor keys, a wait cycle (NOP) may be required before clearing the KEYACC bit.
5. If all data written match the backdoor keys, the MCU is unsecured and the SEC[1:0] bits in the NVOPT register are forced to an unsecured state.

The backdoor key access sequence is monitored by an internal security state machine. An illegal operation during the backdoor key access sequence causes the security state machine to lock, leaving the MCU in the secured state. A reset of the MCU causes the security state machine to exit the lock state and allows a new backdoor key access sequence to be attempted. The following operations during the backdoor key access sequence lock the security state machine:

1. If any of the keys written does not match the backdoor keys programmed in the flash array.
2. If the keys are written in the wrong sequence.
3. If any of the keys written are all 0's or all 1's.
4. If the KEYACC bit does not remain set while the keys are written.

5. If any of the keys are written on successive MCU clock cycles.

6. Executing a STOP instruction before all keys have been written.

After the backdoor keys have been correctly matched, the MCU is unsecured. After the MCU is unsecured, the flash security byte can be programmed to the unsecure state, if desired.

In the unsecure state, you have full control of the contents of the backdoor keys by programming the associated addresses in the flash configuration field (see Section 4.2.1, "Flash Module Reserved Memory Locations").

The security as defined in the flash security byte is not changed by using the backdoor key access sequence to unsecure. The stored backdoor keys are unaffected by the backdoor key access sequence. After the next reset of the MCU, the security state of the flash module is determined by the flash security byte. The backdoor key access sequence has no effect on the program and erase protections defined in the flash protection register (FPROT).

It is not possible to unsecure the MCU by using the backdoor key access sequence in background debug mode (BDM).

4.5.6 Resets

4.5.6.1 Flash Reset Sequence

On each reset, the flash module executes a reset sequence to hold CPU activity while reading the following resources from the flash block:

- MCU control parameters (see Section 4.2.1)
- Flash protection byte (see Section 4.2.1 and Section 4.4.2.4)
- Flash nonvolatile byte (see Section 4.2.1)
- Flash security byte (see Section 4.2.1 and Section 4.4.2.2)

4.5.6.2 Reset While Flash Command Active

If a reset occurs while any flash command is in progress, that command is immediately aborted. The state of the flash array address being programmed or the sector/block being erased is not guaranteed.

4.5.6.3 Program and Erase Times

Before any program or erase command can be accepted, the flash clock divider (FCDIV) must be written to set the internal clock for the flash module to a frequency (f_{FCLK}) between 150 kHz and 200 kHz.

If the initial flash event is a mass erase and verify from BDM, then CSR3[31:24] must be loaded before the XCSR is written to initiate the erase and verify. The data in the XCSR and CSR3 is then loaded into the flash's FCDIV register. (See Section 18.3.4, "Configuration/Status Register 3 (CSR3)"). However, if the first flash event is executed by the processor directly, the flash's FCDIV register is written directly, and the XCSR and CSR3 are not involved.

One period of the resulting clock ($1/f_{FCLK}$) is used by the command processor to time program and erase pulses. An integer number of these timing pulses are used by the command processor to complete a program or erase command.

Program and erase times are given in the *MCF51QE128 Data Sheet*, order number MCF51QE128DS.

4.6   Security

The MCF51QE128/64/32 includes circuitry to prevent unauthorized access to the contents of flash and RAM memory. When security is engaged, BDM access is restricted to the upper byte of the debug XCSR, CSR2, and CSR3 registers. RAM, flash memory, peripheral registers, and most of the CPU register set are not available via BDM. Programs executing from internal memory have normal access to all MCU memory locations and resources.

Security is engaged or disengaged based on the state of two nonvolatile register bits (SEC01, SEC00) in the FOPT register. During reset, the contents of the nonvolatile location, NVOPT, are copied from flash into the working FOPT register in high-page register space. Engage security by programming the NVOPT location. This can be done at the same time the flash memory is programmed. The 1:1 state disengages security and the other three combinations engage security. Security is implemented differently than on the pin-compatible MC9S08QE128/64/32 family of devices. This is a result of differences inherent in the S08 and ColdFire MCU architectures.

Upon exiting reset, the XCSR[SEC] bit in the ColdFire CPU is set if the device is secured, cleared otherwise.

You can allow or disallow a security unlocking mechanism through an 8-byte backdoor security key. The security key can be written by the CPU executing from internal memory. It cannot be entered without the cooperation of a secure user program. The procedure for this is detailed in Section 4.5.5.1, "Unsecuring the MCU using Backdoor Key Access".

Development tools unsecure devices via an alternate BDM-based methodology shown in Figure 4-15. Because $\overline{\text{RESET}}$ and BKGD pins can be reprogrammed via software, a power-on-reset is required to be absolutely certain of obtaining control of the device via BDM, which is a required prerequisite for clearing security. Other methods (outlined in red in Figure 4-15) can also be used, but may not work under all circumstances.

The flowchart contains the following elements:

- Secure state unknown/unpowered
- Hold BKGD=0, apply power, wait N+16 cycles for POR to deassert
- Secure state unknown, CPU halted, FEI 10 MHz clock, SYNC required
- SYNC
- Secure state unknown, CPU halted, FEI 10 MHz clock, synchronized to debugger
- Read XCSR
- Set PRDIV8 and clock divider fields in CSR3 (XCSR[31:24] = 0x87)
- Write XCSR[31:24] = 0x87 to initiate erase/verify of flash memory **Note:** This write is required
- Is XCSR[25] cleared (erase/verify complete)? → No loops back; Yes continues
- On-chip flash is erased and unsecure
- Write CSR2[25:24]=11 to initiate BDM reset to halt or write CSR2[25:24]=01 to initiate BDM reset to run
- Delay 'TBD' cycles
- Device is unsecure

Side notes:

N = number of cycles for SIM to release internal reset. Adder of 16 imposed by ColdFire core.

BKGD=0 during reset ensures that ENBDM comes up '1'.

FLL enabled, internal reference (FEI) at 10MHz is reset default for the ICS.

XCSR[31:24] ≠ 1000_01*x*1 → STOP Error condition check code or device.

XCSR[25] = 0 → STOP Already unsecured

Ways to enter BDM halt mode:

1. BKGD=0 during POR
2. BKGD=0 during external reset
3. BKGD=0 during BDM reset
4. BFHBR=1 during BDM reset
5. Issue BACKGROUND command via BDM interface
6. HALT instruction
7. BDM breakpoint
8. ColdFire fault-on-fault

Of these, only method 1 is guaranteed to work under all circumstances because of the ability to program different functions on the BKGD package pin.

1. The last three steps are optional, but recommended.

Figure 4-15. Procedure for Clearing Security on MCF51QE128/64/32 via the BDM Port

Chapter 5
Resets, Interrupts, and General System Control

5.1   Introduction

This section discusses basic reset and interrupt mechanisms and the various sources of reset and interrupt on the MCF51QE128/64/32. Some interrupt sources from peripheral modules are discussed in greater detail within other sections of this document. This section gathers basic information about all reset and interrupt sources in one place for easy reference. A few reset and interrupt sources, including the computer operating properly (COP) watchdog, are not part of on-chip peripheral systems with their own chapters.

5.2   Features

Reset and interrupt features include:

- Multiple sources of reset for flexible system configuration and reliable operation
- System reset status (SRS) register to indicate source of most recent reset
- Separate interrupt vector for most modules (reduces polling overhead)

5.3   Microcontroller Reset

Resetting the MCU provides a way to start processing from a known set of initial conditions. When the ColdFire processor exits reset, it fetches initial 32-bit values for the supervisor stack pointer and program counter from locations 0x(00)00_0000 and 0x(00)00_0004 respectively. On-chip peripheral modules are disabled and I/O pins are initially configured as general-purpose high-impedance inputs with pull-up devices disabled.

The MCF51QE128/64/32 has the following sources for reset:

- Power-on reset (POR)
- External pin reset (PIN)
- Computer operating properly (COP) timer
- Illegal opcode detect (ILOP)
- Illegal address detect (ILAD)
- Low-voltage detect (LVD)
- Background debug forced reset

Each of these sources, with the exception of the background debug forced reset, has an associated bit in the system reset status register (SRS).

5.3.1 Computer Operating Properly (COP) Watchdog

The COP watchdog forces a system reset when the application software fails to execute as expected. To prevent a system reset from the COP timer (when it is enabled), application software must reset the COP counter periodically. If the application program gets lost and fails to reset the COP counter before it times out, a system reset is generated to force the system back to a known starting point.

After any reset, the SOPT1[COPE] bit is set enabling the COP watchdog (see Section 5.7.3, "System Options Register 1 (SOPT1)," for additional information). If the COP watchdog is not used in an application, it can be disabled by clearing COPE. The COP counter is reset by writing any value to the address of SRS. This write does not affect the data in the read-only SRS. Instead, the act of writing to this address is decoded and sends a reset signal to the COP counter.

The SOPT2[COPCLKS] bit selects the clock source used for the COP timer (see Section 5.7.4, "System Options Register 2 (SOPT2)," for additional information). The clock source options are either the bus clock or an internal 1-kHz clock source. With each clock source, there is an associated short and long time-out controlled by the SOPT1[COPT] bit. Table 5-1 summaries the control functions of the COPCLKS and COPT bits. The COP watchdog defaults to operation from the 1-kHz clock source and the associated long time-out (2^8 cycles).

Table 5-1. COP Configuration Options

Control Bits		Clock Source	COP Overflow Count
COPCLKS	COPT		
0	0	~1 kHz	2^5 cycles (32 ms)[1]
0	1	~1 kHz	2^8 cycles (256 ms)[1]
1	0	Bus	2^{13} cycles
1	1	Bus	2^{18} cycles

[1] Values are shown in this column based on t_{LPO} = 1 ms.

Write to the write-once SOPT1[1] and SOPT2 registers during reset initialization to lock in the settings, even if the application uses the default reset settings of COPE, COPCLKS, and COPT. That way, they cannot be changed accidentally if the application program gets lost. The initial writes to SOPT1 and SOPT2 reset the COP counter.

The write to SRS that services (clears) the COP counter must not be placed in an interrupt service routine (ISR) because the ISR could continue to be executed periodically even if the main application program fails.

In the CPU halt state, the COP counter does not increment.

When the bus clock source is selected, the COP counter does not increment while the system is in stop mode. The COP counter resumes as soon as the MCU exits stop mode.

When the 1-kHz clock source is selected, the COP counter is re-initialized to zero upon entry to stop mode. The COP counter begins from zero after the MCU exits stop mode.

---

1. The SOPT1[WAITE] bit can be written multiple times. Other bits are write-once.

5.3.2 Illegal Operation Reset

By default, the V1 ColdFire core generates a MCU reset when attempting to execute an illegal instruction (except for the ILLEGAL opcode), illegal line-A instruction, illegal line-F instruction, or a supervisor instruction while in user mode (privilege violation). The user may set CPUCR[IRD] to generate the appropriate exception instead of forcing a reset.

NOTE

The attempted execution of the STOP instruction with SOPT[STOPE, WAITE] cleared is treated as an illegal instruction.

The attempted execution of the HALT instruction with XCSR[ENBDM] cleared is treated as an illegal instruction.

5.3.3 Illegal Address Reset

By default, the V1 ColdFire core generates a MCU reset when detecting an address error, bus error termination, RTE format error, or fault-on-fault condition. The user may set CPUCR[ARD] to generate the appropriate exception instead of forcing a reset, or simply halt the processor in response to the fault-on-fault condition.

5.4 Interrupts and Exceptions

The interrupt architecture of ColdFire utilizes a 3-bit encoded interrupt priority level sent from the interrupt controller to the core, providing seven levels of interrupt requests. Level seven represents the highest priority interrupt level, while level one is the lowest priority. For more information on exception processing, see Chapter 8, "Interrupt Controller (CF1_INTC)".

5.4.1 External Interrupt Request (IRQ) Pin

External interrupts are managed by the IRQ status and control register, IRQSC. When the IRQ function is enabled, synchronous logic monitors the pin for edge-only or edge-and-level events. When the MCU is in stop mode and system clocks are shut down, a separate asynchronous path is used, so the IRQ pin (if enabled) can wake the MCU.

5.4.1.1 Pin Configuration Options

The IRQ pin enable (IRQPE) control bit in IRQSC must be set for the IRQ pin to act as the interrupt request (IRQ) input. As an IRQ input, you can choose the polarity of edges or levels detected (IRQEDG), whether the pin detects edges-only or edges and levels (IRQMOD) and whether an event causes an interrupt or only sets the IRQF flag that can be polled by software (IRQIE).

The IRQ pin, when enabled, defaults to use an internal pull device (IRQPDD = 0), configured as a pull-up or pull-down depending on the polarity chosen. If you want to use an external pull-up or pull-down, the IRQPDD can be set to turn off the internal device.

NOTE

This pin does not contain a clamp diode to V_{DD} and should not be driven above V_{DD}.

NOTE

The voltage measured on the internally pulled up $\overline{RESET}$ pin is not pulled to V_{DD}. The internal gates connected to this pin are pulled to V_{DD}. The $\overline{RESET}$ pullup should not be used to pull up components external to the MCU.

5.4.1.2 Edge and Level Sensitivity

The IRQMOD control bit reconfigures the detection logic so it detects edge events and pin levels. In the edge and level detection mode, the IRQF status flag becomes set when an edge is detected (when the IRQ pin changes from the deasserted to the asserted level), but the flag is continuously set (and cannot be cleared) as long as the IRQ pin remains at the asserted level.

5.4.1.3 External Interrupt Initialization

When the IRQ pin is first enabled, it is possible to get a false interrupt flag. To prevent a false interrupt request during IRQ initialization, the user should do the following:

1. Mask interrupts by clearing IRQSC[IRQIE].
2. Select the pin polarity by setting the appropriate IRQSC[IRQEDG] bits.
3. If using internal pull-up/pull-down device, clear IRQSC[IRQPDD].
4. Enable the IRQ pin by setting IRQSC[IRQPE].
5. Write to IRQSC[IRQACK] to clear any false interrupts.
6. Set IRQSC[IRQIE] to enable interrupts.

5.5 Low-Voltage Detect (LVD) System

The MCF51QE128/64/32 includes a system to guard against low voltage conditions to protect memory contents and control MCU system states during supply voltage variations. The system is comprised of a power-on reset (POR) circuit and a LVD circuit with a user-selectable trip voltage, high (V_{LVDH}) or low (V_{LVDL}). The LVD circuit is enabled when the SPMSC1[LVDE] bit is set and the trip voltage is selected by the SPMSC3[LVDV] bit. The LVD is disabled upon entering stop2 or stop3 modes unless the LVDSE bit is set. If LVDE and LVDSE are set when the STOP instruction is processed, the device enters stop4 mode. The LVD can be left enabled in this mode.

5.5.1 Power-On Reset Operation

When power is initially applied to the MCU or the supply voltage drops below the power-on reset re-arm voltage level, V_{POR}, the POR circuit causes a reset condition. As the supply voltage rises, the LVD circuit holds the MCU in reset until the supply has risen above the LVD low threshold, V_{LVDL}. The SRS[POR,LVD] bits are set following a POR.

5.5.2   LVD Reset Operation

The LVD can be configured to generate a reset upon detection of a low voltage condition by setting LVDRE to 1. The low voltage detection threshold is determined by the LVDV bit. After an LVD reset has occurred, the LVD system holds the MCU in reset until the supply voltage has risen above the low voltage detection threshold. SRS[LVD] is set following an LVD reset or POR.

5.5.3   LVD Interrupt Operation

When a low voltage condition is detected and the LVD circuit is configured using SPMSC1 for interrupt operation (LVDE set, LVDIE set, and LVDRE clear), SPMSC1[LVDF] is set and an LVD interrupt request occurs. The LVDF bit is cleared by writing a 1 to the LVDACK bit in SPMSC1.

5.5.4   Low-Voltage Warning (LVW) Interrupt Operation

The LVD system has a low voltage warning flag (LVWF) to indicate the supply voltage is approaching, but is above, the LVD voltage. The LVW also has an interrupt associated with it, enabled by setting the SPMSC3[LVWIE] bit. If enabled, an LVW interrupt request occurs when the LVWF is set. LVWF is cleared by writing a 1 to the SPMSC3[LVWACK] bit. There are two user-selectable trip voltages for the LVW, one high (V_{LVWH}) and one low (V_{LVWL}). The trip voltage is selected by SPMSC3[LVWV] bit.

5.6   Peripheral Clock Gating

The MCF51QE128/64/32 includes a clock gating system to manage the bus clock sources to the individual peripherals. Using this system, you can enable or disable the bus clock to each of the peripherals at the clock source, eliminating unnecessary clocks to peripherals which are not in use. This reduces the overall run and wait mode currents.

Out of reset, all peripheral clocks are enabled. For lowest possible run or wait currents, software should disable the clock source to any peripheral not in use. The actual clock is enabled or disabled immediately following the write to the clock gating control registers (SCGC1, SCGC2). Any peripheral with a gated clock cannot be used unless its clock is enabled. Writing to the registers of a peripheral with a disabled clock has no effect.

NOTE

Software should disable the peripheral before disabling the clocks to the peripheral. After clocks are re-enabled to a peripheral, the peripheral registers need to be re-initialized by software.

In stop modes, the bus clock is disabled for all gated peripherals, regardless of the settings in the SCGC1 and SCGC2 registers.

5.7   Reset, Interrupt, and System Control Registers and Control Bits

One 8-bit register in the direct page register space and eight 8-bit registers in the high-page register space are related to reset and interrupt systems.

Refer to Chapter 4, "Memory", for the absolute address assignments for all registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

Some control bits in the SOPT1 and SPMSC2 registers are related to modes of operation. Although brief descriptions of these bits are provided here, the related functions are discussed in greater detail in Chapter 3, "Modes of Operation".

5.7.1   Interrupt Pin Request Status and Control Register (IRQSC)

This direct page register includes status and control bits which are used to configure the IRQ function, report status, and acknowledge IRQ events.

	7	6	5	4	3	2	1	0
R	0	IRQPDD	IRQEDG	IRQPE	IRQF	0	IRQIE	IRQMOD
W						IRQACK		
Reset	0	0	0	0	0	0	0	0
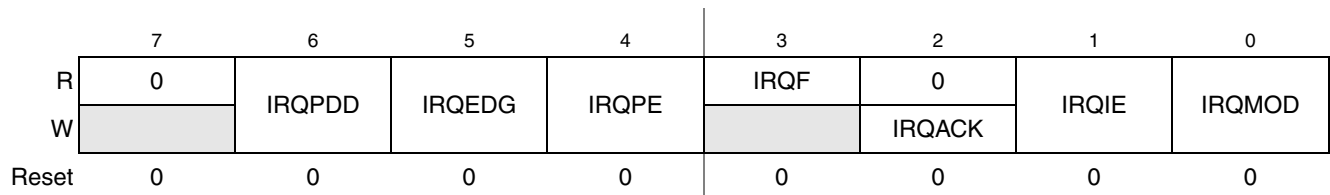
Figure 5-1. Interrupt Request Status and Control Register (IRQSC)

Table 5-2. IRQSC Register Field Descriptions

Field	Description
7	Reserved, should be cleared.
6 IRQPDD	Interrupt Request (IRQ) Pull Device Disable. This read/write control bit is used to disable the internal pull-up/pull-down device when the IRQ pin is enabled (IRQPE = 1) allowing for an external device to be used. 0 IRQ pull device enabled if IRQPE = 1. 1 IRQ pull device disabled if IRQPE = 1.
5 IRQEDG	Interrupt Request (IRQ) Edge Select. This read/write control bit selects the polarity of edges or levels on the IRQ pin that cause IRQF to be set. The IRQMOD control bit determines whether the IRQ pin is sensitive to edges and levels or only edges. When IRQEDG is set and the internal pull device is enabled, the pull-up device is reconfigured as an optional pull-down device. 0 IRQ is falling edge or falling edge/low-level sensitive. 1 IRQ is rising edge or rising edge/high-level sensitive.
4 IRQPE	IRQ Pin Enable. This read/write control bit enables the IRQ pin function. When this bit is set, the IRQ pin can be used as an external interrupt request. 0 IRQ pin function is disabled. 1 IRQ pin function is enabled.
3 IRQF	IRQ Flag. This read-only status bit indicates when an interrupt request event has occurred. 0 No IRQ request. 1 IRQ event detected.
2 IRQACK	IRQ Acknowledge. This write-only bit acknowledges interrupt request events (write 1 to clear IRQF). Writing 0 has no meaning or effect. Reads always return 0. If edge-and-level detection is selected (IRQMOD = 1), IRQF cannot be cleared while the IRQ pin remains at its asserted level.

Table 5-2. IRQSC Register Field Descriptions (continued)

Field	Description
1 IRQIE	IRQ Interrupt Enable. This read/write control bit determines whether IRQ events generate an interrupt request. 0 Interrupt request when IRQF set is disabled (use polling). 1 Interrupt requested when IRQF is set.
0 IRQMOD	IRQ Detection Mode. This read/write control bit selects edge-only detection or edge-and-level detection. The IRQEDG control bit determines the polarity of edges and levels detected as interrupt request events. See Section 5.4.1.2, "Edge and Level Sensitivity" for more details. 0 IRQ event on falling edges or rising edges only. 1 IRQ event on falling edges and low levels or on rising edges and high levels.

5.7.2   System Reset Status Register (SRS)

This high page register includes read-only status flags to indicate the source of the most recent reset. When a debug host forces reset by setting CSR2[BDFR], none of the status bits in SRS are set. Writing any value to this register address clears the COP watchdog timer without affecting the contents of this register. The reset state of these bits depends on what caused the MCU to reset.

	7	6	5	4	3	2	1	0
R	POR	PIN	COP	ILOP	ILAD	0	LVD	0
W	Writing any value to SRS address clears COP watchdog timer.							
POR:	1	0	0	0	0	0	1	0
LVD:	u	0	0	0	0	0	1	0
Any other reset:	0	Note[1]	Note[1]	Note[1]	Note[1]	0	0	0

[1] Any of these reset sources active at the time of reset entry causes the corresponding bit(s) to be set; bits corresponding to sources not active at the time of reset entry are cleared.

Figure 5-2. System Reset Status (SRS)

Table 5-3. SRS Register Field Descriptions

Field	Description
7 POR	Power-On Reset. Reset was caused by the power-on detection logic. Because the internal supply voltage was ramping up at the time, the low-voltage reset (LVD) status bit is also set to indicate that the reset occurred while the internal supply was below the LVD threshold. 0 Reset not caused by POR. 1 POR caused reset.
6 PIN	External Reset Pin. Reset was caused by an active-low level on the external reset pin. 0 Reset not caused by external reset pin. 1 Reset came from external reset pin.
5 COP	Computer Operating Properly (COP) Watchdog. Reset was caused by the COP watchdog timer timing out. This reset source is blocked if COPE is cleared. 0 Reset not caused by COP timeout. 1 Reset caused by COP timeout.

Table 5-3. SRS Register Field Descriptions (continued)

Field	Description
4 ILOP	Illegal Opcode. Reset was caused by an attempt to execute an unimplemented or illegal opcode. This includes any illegal instruction (except the ILLEGAL (0x4AFC) opcode) or a privilege violation (execution of a supervisor instruction in user mode. The STOP instruction is considered illegal if SOPT1[STOPE,WAITE] are cleared. The HALT instruction is considered illegal if the BDM interface is disabled (XCSR[ENBDM] = 0). All illegal opcode resets are enabled when CPUCR[IRD] is cleared. If CPUCR[IRD] is set, then the appropriate processor exception is generated instead of a reset. 0 Reset not caused by an illegal opcode. 1 Reset caused by an illegal opcode.
3 ILAD	Illegal Address. Reset was caused by the processor's attempted access of an illegal address in the memory map, an address error, an RTE format error, or a fault-on-fault condition. All the illegal address resets are enabled when CPUCR[ARD] is cleared. When CPUCR[ARD] is set, the appropriate processor exception is generated instead of a reset, or if a fault-on-fault condition is reached, the processor simply halts. 0 Reset not caused by an illegal access. 1 Reset caused by an illegal access.
2	Reserved, should be cleared.
1 LVD	Low Voltage Detect. If LVDRE is set and the supply drops below the LVD trip voltage, an LVD reset occurs. This bit is also set by POR. 0 Reset not caused by LVD trip or POR. 1 Reset caused by LVD trip or POR.
0	Reserved, should be cleared.

5.7.3   System Options Register 1 (SOPT1)

All SOPT1 bit fields, except WAITE, are write-once. Therefore for the write-once bits, only the first write after reset is honored. Any subsequent attempt to write to these bit fields (intentionally or unintentionally) is ignored to avoid accidental changes to these sensitive settings. All bit fields may be read at any time and WAITE is write-anytime. SOPT1 should be written during the reset initialization program to set the desired controls even if the desired settings are the same as the reset settings.

	7	6	5	4	3	2	1	0
R	COPE	COPT	STOPE	WAITE	0	RSTOPE	BKGDPE	RSTPE
W								
Reset:	1	1	0	1	0	u[1]	1	u[1]
POR:	1	1	0	1	0	0	1	0
LVR:	1	1	0	1	0	0	1	0

[1] u = unaffected

Figure 5-3. System Options Register 1 (SOPT1)

Table 5-4. SOPT1 Register Field Descriptions

Field	Description
7 COPE	COP Watchdog Enable. This write-once bit selects whether the COP watchdog is enabled. 0 COP watchdog timer disabled. 1 COP watchdog timer enabled (force reset on timeout).
6 COPT	COP Watchdog Timeout. This write-once bit selects the timeout period of the COP. COPT along with SOPT2[COPCLKS] defines the COP timeout period. 0 Short timeout period selected. 1 Long timeout period selected.
5 STOPE	Stop Mode Enable. This write-once bit is used to enable stop mode. If stop and wait modes are disabled and a user program attempts to execute a STOP instruction, an illegal opcode reset may be generated depending on CPUCR[IRD].
4 WAITE	Wait Mode Enable. This write-anytime bit is used to enable wait mode. If stop and wait modes are disabled and a user program attempts to execute a STOP instruction, an illegal opcode reset may be generated depending on CPUCR[IRD]. If this bit is set, the value of STOPE is ignored when the STOP instruction is executed. **Note:** The user must execute a NOP instruction following any change to WAITE to allow the updated register bit time to propagate to the processor.
3	Reserved, should be cleared.
2 RSTOPE	$\overline{\text{RSTO}}$ Pin Enable. When set, this write-once bit enables the PTC4/RGPIO12/TPM3CH4/$\overline{\text{RSTO}}$ pin to function as $\overline{\text{RSTO}}$ When clear, the pin functions as one of its alternative functions. This pin defaults to its I/O port function following an MCU POR. When RSTOPE is set, an internal pullup device is enabled on $\overline{\text{RSTO}}$. The pin remains in this mode until the next power-on-reset.
1 BKGDPE	Background Debug Mode Pin Enable. When set, this write-once bit enables the PTA4/ACMP1O/BKGD/MS pin to function as BKGD/MS. When clear, the pin functions as one of its output-only alternative functions. This pin defaults to the BKGD/MS function following any MCU reset.
0 RSTPE	$\overline{\text{RESET}}$ Pin Enable. This write-once bit when set enables the PTA5/IRQ/TPM1CLK/$\overline{\text{RESET}}$ pin to function as $\overline{\text{RESET}}$. When clear, the pin functions as one of its input-only alternative functions. This pin defaults to its I/O port function following an MCU POR. When RSTPE is set, an internal pullup device is enabled on $\overline{\text{RESET}}$.

5.7.4 System Options Register 2 (SOPT2)

This high page register contains bits to configure MCU specific features on the MCF51QE128/64/32 devices.

	7	6	5	4	3	2	1	0
R	COPCLKS[1]	0	0	0	SPI1PS	ACIC2	IICPS	ACIC1
W								
Reset:	0	0	0	0	0	0	0	0
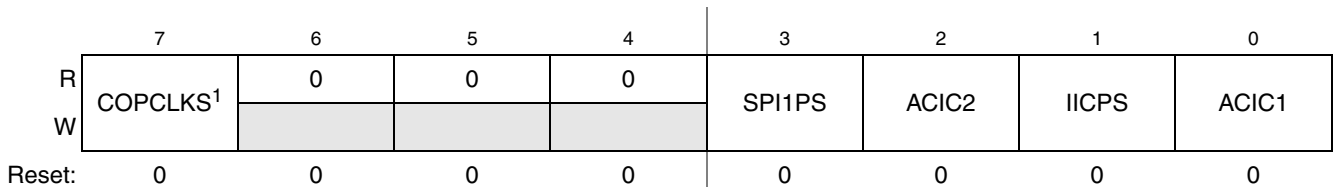
[1] This bit can be written only one time after reset. Additional writes are ignored.

Figure 5-4. System Options Register 2 (SOPT2)

Table 5-5. SOPT2 Register Field Descriptions

Field	Description
7 COPCLKS	COP Watchdog Clock Select. This write-once bit selects the clock source of the COP watchdog. 0 Internal 1-kHz clock is source to COP. 1 Bus clock is source to COP.
6–4	Reserved, should be cleared.
3 SPI1PS	SPI1 Pin Select. This bit selects the location of the SPSCLK1, MOSI1, MISO1, and SS1 pins of the SPI1 module. <table><tr><th>SPI1PS</th><th>SPSCLK1</th><th>MOSI1</th><th>MISO1</th><th>SS1</th></tr><tr><td>0</td><td>PTB2</td><td>PTB3</td><td>PTB4</td><td>PTB5</td></tr><tr><td>1</td><td>PTE0</td><td>PTE1</td><td>PTE2</td><td>PTE3</td></tr></table>
2 ACIC2	Analog Comparator 2 to Input Capture Enable. This bit connects the output of ACMP2 to TPM2 input channel 0. See Chapter 10, "Analog Comparator 3V (ACMPVLPV1)," and Chapter 17, "Timer/Pulse-Width Modulator (S08TPMV3)," for more details on this feature. 0 ACMP2 output not connected to TPM2 input channel 0. 1 ACMP2 output connected to TPM2 input channel 0.
1 IIC1PS	IIC1 Pin Select. This bit selects the location of the SDA1 and SCL1 pins of the IIC1 module. <table><tr><th>IIC1PS</th><th>SDA1</th><th>SCL1</th></tr><tr><td>0</td><td>PTA2</td><td>PTA3</td></tr><tr><td>1</td><td>PTB6</td><td>PTB7</td></tr></table>
0 ACIC1	Analog Comparator 1 to Input Capture Enable. This bit connects the output of ACMP1 to TPM1 input channel 0. See Chapter 10, "Analog Comparator 3V (ACMPVLPV1)," and Chapter 17, "Timer/Pulse-Width Modulator (S08TPMV3)," for more details on this feature. 0 ACMP output not connected to TPM1 input channel 0. 1 ACMP output connected to TPM1 input channel 0.

5.7.5   System Device Identification Register (SDIDH, SDIDL)

These high page read-only registers are included so host development systems can identify the ColdFire derivative. This allows the development software to recognize where specific memory blocks, registers, and control bits are located in a target MCU.

Additional configuration information about the ColdFire core and memory system is loaded into the 32-bit D0 and D1 registers at reset. This information can be stored into memory by the system startup code for later use by configuration-sensitive application code. See Chapter 7, "ColdFire Core," for more information.

	7	6	5	4	3	2	1	0
R						ID[11:8]		
W								
Reset:	—	—	—	—	1	1	0	0

Figure 5-5. System Device Identification Register High (SDIDH)

Table 5-6. SDIDH Register Field Descriptions

Field	Description
7–4 Reserved	Reserved. Reading these bits results in an indeterminate value; writes have no effect.
3–0 ID[11:8]	Part Identification Number. Each derivative in the ColdFire family has a unique identification number. The MCF51QE128/64/32 are hard coded to the value 0xC15. See also ID bits in Table 5-7.

	7	6	5	4	3	2	1	0
R				ID[7:0]				
W								
Reset:	0	0	0	1	0	1	0	1
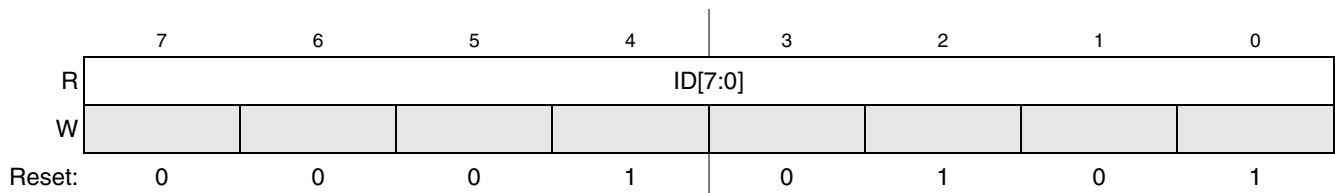
Figure 5-6. System Device Identification Register Low (SDIDL)

Table 5-7. SDIDL Register Field Descriptions

Field	Description
7–0 ID[7:0]	Part Identification Number. Each derivative in the ColdFire family has a unique identification number. The MCF51QE128/64/32 are hard coded to the value 0xC15. See also ID bits in Table 5-6.

5.7.6 System Power Management Status and Control 1 Register (SPMSC1)

This high page register contains status and control bits to support the low voltage detect function, and it enables the bandgap voltage reference for use by the ADC module. To configure the low voltage detect trip voltage, see Table 5-10 for the LVDV bit description in SPMSC3.

SPMSC1 is not reset when exiting from stop2.

	7	6	5	4	3	2	1	0
R	LVDF	0	LVDIE	LVDRE[2]	LVDSE	LVDE[2]	0[1]	BGBE
W		LVDACK						
Reset:	0	0	0	1	1	1	0	0
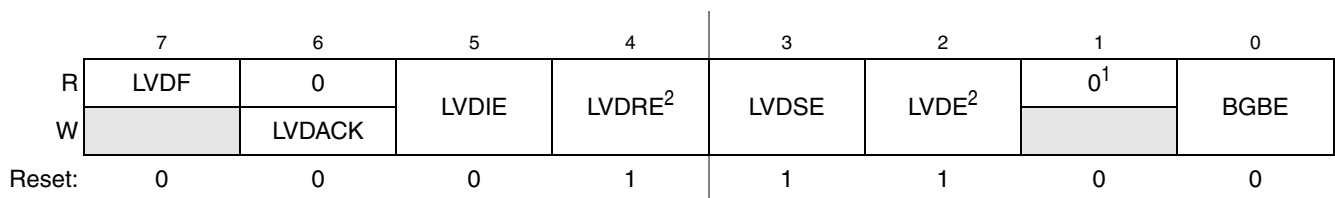
[1] Bit 1 is a reserved bit that must always be written to 0.

[2] This bit can be written only one time after reset. Additional writes are ignored.

Figure 5-7. System Power Management Status and Control 1 Register (SPMSC1)

Table 5-8. SPMSC1 Register Field Descriptions

Field	Description
7 LVDF	Low-Voltage Detect Flag. If LVDE is set, this read-only status bit indicates a low-voltage detect event.
6 LVDACK	Low-Voltage Detect Acknowledge. This write-only bit is used to acknowledge low voltage detection errors (write 1 to clear LVDF). Reads always return 0.
5 LVDIE	Low-Voltage Detect Interrupt Enable. This bit enables hardware interrupt requests for LVDF. 0 Hardware interrupt disabled (use polling). 1 Request a hardware interrupt when LVDF is set.
4 LVDRE	Low-Voltage Detect Reset Enable. This write-once bit enables LVDF events to generate a hardware reset (provided LVDE = 1). 0 LVDF does not generate hardware resets. 1 Force an MCU reset when LVDF is set.
3 LVDSE	Low-Voltage Detect Stop Enable. If LVDE is set, this read/write bit determines whether the low-voltage detect function operates when the MCU is in stop mode. 0 Low-voltage detect disabled during stop mode. 1 Low-voltage detect enabled during stop mode.
2 LVDE	Low-Voltage Detect Enable. This write-once bit enables low-voltage detect logic and qualifies the operation of other bits in this register. 0 LVD logic disabled. 1 LVD logic enabled.
1	Reserved, must be cleared.
0 BGBE	Bandgap Buffer Enable. This bit enables an internal buffer for the bandgap voltage reference for use by the ADC module on one of its internal channels or as a voltage reference for ACMP module. 0 Bandgap buffer disabled. 1 Bandgap buffer enabled.

5.7.7 System Power Management Status and Control 2 Register (SPMSC2)

This high page register contains status and control bits to configure the low-power run and wait modes as well as configure the stop mode behavior of the MCU. See Section 3.7.2, "Low-Power Wait Mode (LPwait)," and Section 3.8, "Stop Modes," for more information.

	7	6	5	4	3	2	1	0
R	LPR	LPRS	LPWUI	0	PPDF	0	PPDE[1]	PPDC
W						PPDACK		
Reset:	0	0	0	0	0	0	1	0

[1] PPDE is a write-once bit that can be used to permanently disable the PPDC bit.

Figure 5-8. System Power Management Status and Control 2 Register (SPMSC2)

SPMSC2 is not reset when exiting from stop2.

Table 5-9. SPMSC2 Register Field Descriptions

Field	Description
7 LPR	Low-Power Regulator Control. The LPR bit controls entry into the low-power run and low-power wait modes in which the voltage regulator is put into standby. This bit cannot be set if PPDC=1. If PPDC and LPR are set in a single write instruction, only PPDC is actually set. LPR is cleared when an interrupt occurs in low-power mode and the LPWUI bit is 1. 0 Low-power run and low-power wait modes are disabled. 1 Low-power run and low-power wait modes are requested.
6 LPRS	Low-Power Regulator Status. This read-only status bit indicates that the voltage regulator has entered into standby for the low-power run or wait mode. 0 The voltage regulator is not currently in standby. 1 The voltage regulator is currently in standby.
5 LPWUI	Low-Power Wake-Up on Interrupt. This bit controls whether or not the voltage regulator exits standby when any active MCU interrupt occurs. 0 The voltage regulator remains in standby on an interrupt. 1 The voltage regulator exits standby on an interrupt. LPR is cleared.
4	Reserved, should be cleared.
3 PPDF	Partial Power-Down Flag. This read-only status bit indicates that the MCU has recovered from stop2 mode. 0 MCU has not recovered from stop2 mode. 1 MCU recovered from stop2 mode.
2 PPDACK	Partial Power-Down Acknowledge. Writing a 1 to PPDACK clears the PPDF bit.
1 PPDE	Partial Power-Down Enable. The write-once PPDE bit can be used to lockout the partial power-down feature. This is a write-once bit. 0 Partial power-down is not enabled. 1 Partial power-down is enabled and controlled via the PPDC bit.
0 PPDC	Partial Power-Down Control. The PPDC bit controls which power-down mode is selected. This bit cannot be set if LPR is set. If PPDC and LPR are set in a single write instruction, only PPDC is actually set. PPDE must be set for PPDC to be set. 0 Stop3 low-power mode enabled. 1 Stop2 partial power-down mode enabled. There are also restrictions on LVDE and LVDSE. See Table 3-1 for details.

5.7.8 System Power Management Status and Control 3 Register (SPMSC3)

This high page register reports the status of the low voltage warning function and to select the low voltage detect trip voltage. SPMSC3 is not reset when exiting from stop2.

	7	6	5	4	3	2	1	0
R	LVWF	0	LVDV	LVWV	LVWIE	0	0	0
W		LVWACK						
POR:	0[1]	0	0	0	0	0	0	0
LVR:	0[1]	0	U	U	0	0	0	0
Any other reset:	0[1]	0	U	U	0	0	0	0

[1] LVWF is set when V_{Supply} transitions below the trip point or after reset and V_{Supply} is already below V_{LVW}.

Figure 5-9. System Power Management Status and Control 3 Register (SPMSC3)

Table 5-10. SPMSC3 Register Field Descriptions

Field	Description
7 LVWF	Low-Voltage Warning Flag. The LVWF bit indicates the low voltage warning status. 0 Low voltage warning not present. 1 Low voltage warning is present or was present.
6 LVWACK	Low-Voltage Warning Acknowledge. Writing a 1 to LVWACK clears LVWF if a low voltage warning is not present.
5 LVDV	Low-Voltage Detect Voltage Select. The LVDV bit selects the LVD trip point voltage (V_{LVD}). 0 Low trip point selected ($V_{LVD} = V_{LVDL}$). 1 High trip point selected ($V_{LVD} = V_{LVDH}$).
4 LVWV	Low-Voltage Warning Voltage Select. The LVWV bit selects the LVW trip point voltage (V_{LVW}). 0 Low trip point selected ($V_{LVW} = V_{LVWL}$). 1 High trip point selected ($V_{LVW} = V_{LVWH}$).
3 LVWIE	Low-Voltage Warning Interrupt Enable. This bit enables hardware interrupt requests for LVWF. 0 Hardware interrupt disabled (use polling). 1 Request a hardware interrupt when LVWF is set.
2–0	Reserved, should be cleared.

Table 5-11. LVD and LVW Trip Point Typical Values[1]

LVDV:LVWV	LVW Trip Point	LVD Trip Point
00	$V_{LVWL} = 2.15$	$V_{LVDL} = 1.86$
01	$V_{LVWL} = 2.48$	
10 Not Recommended	$V_{LVWL} = 2.15$	$V_{LVDL} = 2.15$
11	$V_{LVWL} = 2.48$	

[1] See the *MCF51QE128 Data Sheet* for minimum and maximum values.

5.7.9 System Clock Gating Control 1 Register (SCGC1)

This high page register contains control bits to enable or disable the bus clock to the TPMx, ADC, IICx, and SCIx modules. Gating off the clocks to unused peripherals reduces the MCU's run and wait currents. See Section 5.6, "Peripheral Clock Gating," for more information.

	7	6	5	4	3	2	1	0
R	TPM3	TPM2	TPM1	ADC	IIC2	IIC1	SCI2	SCI1
W								
Reset:	1	1	1	1	1	1	1	1
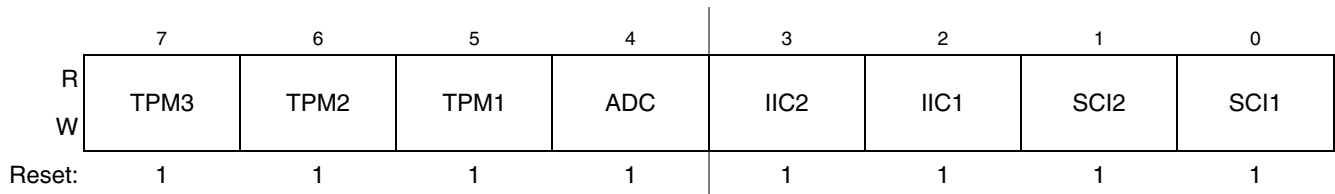
Figure 5-10. System Clock Gating Control 1 Register (SCGC1)

Table 5-12. SCGC1 Register Field Descriptions

Field	Description
7 TPM3	TPM3 Clock Gate Control. This bit controls the clock gate to the TPM3 module. 0 Bus clock to the TPM3 module is disabled. 1 Bus clock to the TPM3 module is enabled.
6 TPM2	TPM2 Clock Gate Control. This bit controls the clock gate to the TPM2 module. 0 Bus clock to the TPM2 module is disabled. 1 Bus clock to the TPM2 module is enabled.
5 TPM1	TPM1 Clock Gate Control. This bit controls the clock gate to the TPM1 module. 0 Bus clock to the TPM1 module is disabled. 1 Bus clock to the TPM1 module is enabled.
4 ADC	ADC Clock Gate Control. This bit controls the clock gate to the ADC module. 0 Bus clock to the ADC module is disabled. 1 Bus clock to the ADC module is enabled.
3 IIC2	IIC2 Clock Gate Control. This bit controls the clock gate to the IIC2 module. 0 Bus clock to the IIC2 module is disabled. 1 Bus clock to the IIC2 module is enabled.
2 IIC1	IIC1 Clock Gate Control. This bit controls the clock gate to the IIC1 module. 0 Bus clock to the IIC1 module is disabled. 1 Bus clock to the IIC1 module is enabled.
1 SCI2	SCI2 Clock Gate Control. This bit controls the clock gate to the SCI2 module. 0 Bus clock to the SCI2 module is disabled. 1 Bus clock to the SCI2 module is enabled.
0 SCI1	SCI1 Clock Gate Control. This bit controls the clock gate to the SCI1 module. 0 Bus clock to the SCI1 module is disabled. 1 Bus clock to the SCI1 module is enabled.

5.7.10 System Clock Gating Control 2 Register (SCGC2)

This high page register contains control bits to enable or disable the bus clock to the IRQ, KBI, ACMP, RTC, and SPIx modules. Gating off the clocks to unused peripherals reduces the MCU's run and wait currents. See Section 5.6, "Peripheral Clock Gating," for more information.

	7	6	5	4	3	2	1	0
R	1	FLS	IRQ	KBI*x*	ACMP*x*	RTC	SPI2	SPI1
W								
Reset:	1	1	1	1	1	1	1	1

Figure 5-11. System Clock Gating Control 2 Register (SCGC2)

Table 5-13. SCGC2 Register Field Descriptions

Field	Description
7	Reserved, must be set.
6 FLS	FTSR Clock Gate Control. This bit controls the bus clock gate to the flash registers. This bit does not affect normal program execution from the flash array. Only the clock to the flash control registers is affected. 0 Bus clock to flash registers is disabled. 1 Bus clock to flash registers is enabled.
5 IRQ	IRQ Clock Gate Control. This bit controls the bus clock gate to the IRQ module. 0 Bus clock to the IRQ module is disabled. 1 Bus clock to the IRQ module is enabled.
4 KBI*x*	KBI Clock Gate Control. This bit controls the clock gate to both of the KBI modules. 0 Bus clock to the KBI modules is disabled. 1 Bus clock to the KBI modules is enabled.
3 ACMP*x*	ACMP Clock Gate Control. This bit controls the clock gate to both of the ACMP modules. 0 Bus clock to the ACMP modules is disabled. 1 Bus clock to the ACMP modules is enabled.
2 RTC	RTC Clock Gate Control. This bit controls the bus clock gate to the RTC module. Only the bus clock is gated; the ICSERCLK and LPOCLK are still available to the RTC. 0 Bus clock to the RTC module is disabled. 1 Bus clock to the RTC module is enabled.
1 SPI2	SPI2 Clock Gate Control. This bit controls the clock gate to the SPI2 module. 0 Bus clock to the SPI2 module is disabled. 1 Bus clock to the SPI2 module is enabled.
0 SPI1	SPI1 Clock Gate Control. This bit controls the clock gate to the SPI1 module. 0 Bus clock to the SPI1 module is disabled. 1 Bus clock to the SPI1 module is enabled.

Get the latest version from freescale.com

Chapter 6
Parallel Input/Output Control

This section explains software controls related to parallel input/output (I/O) and pin control. The MCF51QE128/64 devices have up to nine parallel I/O ports that include a total of 70 I/O pins and one input-only and one output-only pin. See Chapter 2, "Pins and Connections," for more information about pin assignments and external hardware considerations of these pins.

In addition to standard I/O port functionality, ports C and E have set/clear/toggle functions integrated as part of the ColdFire core itself to improve edge resolution on those pins. See Section 6.4, "V1 ColdFire Rapid GPIO Functionality," and Chapter 9, "Rapid GPIO (RGPIO)," for additional details.

Many port pins are shared with on-chip peripherals such as timer systems, communication systems, or keyboard interrupts as shown in Figure 1-1. The peripheral modules have priority over the general-purpose I/O functions so that when a peripheral is enabled, the I/O functions associated with the shared pins may be disabled.

After reset, the shared peripheral functions are disabled and the pins are configured as inputs (PTxDDn = 0). The pin control functions for each pin are configured as follows: slew rate control disabled (PTxSEn = 0), low drive strength selected (PTxDSn = 0), and internal pull-ups disabled (PTxPEn = 0).

NOTE

> Not all general-purpose I/O pins are available on all packages. To avoid extra current drain from floating input pins, the reset initialization routine in the application program must enable on-chip pull-up devices or change the direction of unconnected pins to outputs so the pins do not float.

6.1 Port Data and Data Direction

Reading and writing of parallel I/Os are performed through the port data registers. The direction, input or output, is controlled through the port data direction registers. The parallel I/O port function for an individual pin is illustrated in the block diagram shown in Figure 6-1.

The data direction control bit (PTxDDn) determines whether the output buffer for the associated pin is enabled and also controls the source for port data register reads. The input buffer for the associated pin is always enabled unless the pin is enabled as an analog function or is an output-only pin.

When a shared digital function is enabled for a pin, the output buffer is controlled by the shared function. However, the data direction register bit continues to control the source for reads of the port data register.

When a shared analog function is enabled for a pin, the input and output buffers are disabled. A value of 0 is read for any port data bit where the bit is an input (PTxDDn = 0) and the input buffer is disabled. In general, when a pin is shared with an alternate digital function and an analog function, the analog function has priority. Therefore, if the digital and analog functions are enabled, the analog function controls the pin.

It is good programming practice to write to the port data register before changing the direction of a port pin to become an output. This ensures that the pin is not driven for a short time with an old data value that happened to be in the port data register.
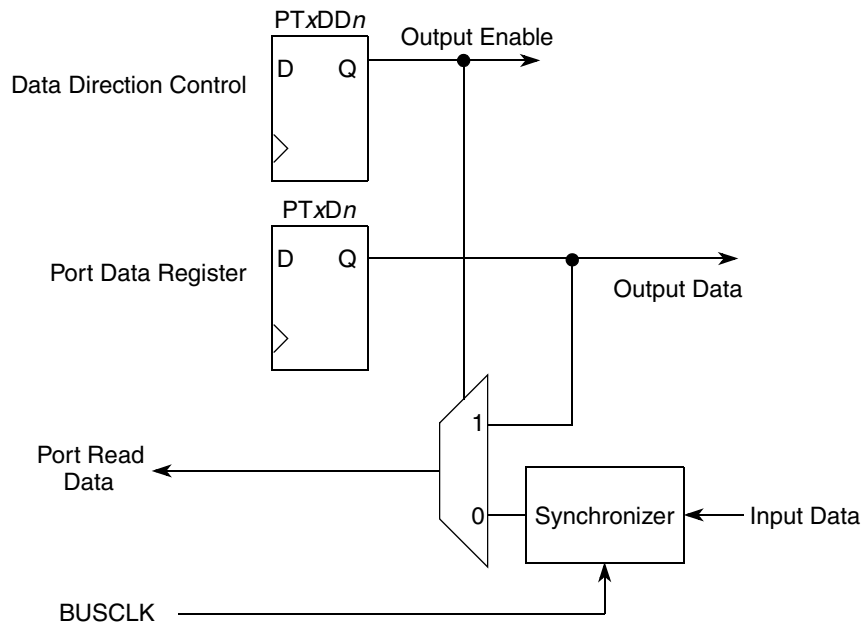
Figure 6-1. Classic Parallel I/O Block Diagram

6.2 Pull-up, Slew Rate, and Drive Strength

A set of high page registers control pull-ups, slew rate, and drive strength for the pins. They may also be used with the peripheral functions on these pins. These registers are associated with the parallel I/O ports, but operate independently of the parallel I/O registers.

6.2.1 Port Internal Pull-up Enable

An internal pull-up device can be enabled for each port pin by setting the corresponding bit in the pull-up enable register (PT*x*PE*n*). The pull-up device is disabled if the pin is configured as an output by the parallel I/O control logic or any shared peripheral function regardless of the state of the corresponding pull-up enable register bit. The pull-up device is also disabled if the pin is controlled by an analog function.

6.2.2 Port Slew Rate Enable

Slew rate control can be enabled for each port pin by setting the corresponding bit in the slew rate control register (PT*x*SE*n*). When enabled, slew control limits the rate at which an output can transition in order to reduce EMC emissions. Slew rate control has no effect on pins configured as inputs.

6.2.3 Port Drive Strength Select

An output pin can be selected to have high output drive strength by setting the corresponding bit in the drive strength select register (PT*x*DSn). When high drive is selected, a pin is capable of sourcing and sinking greater current. Even though every I/O pin can be selected as high drive, ensure that the total current source and sink limits for the MCU are not exceeded. Drive strength selection is intended to affect the DC behavior of I/O pins. However, the AC behavior is also affected. High drive allows a pin to drive a greater load with the same switching speed as a low drive enabled pin into a smaller load. Because of this, the EMC emissions may be affected by enabling pins as high drive.

6.3 Port Data Set, Clear and Toggle Data Registers

The port data set, clear, and toggle registers provide an alternate method for setting and clearing individual port I/O pins within a single port. Only port C and port E have data set, clear, and toggle registers. Figure 6-2 should be contrasted with Figure 6-1 to see the the effects of adding set/clear/toggle functionality to the port cell. SET_Enable, CLR_Enable, and Toggle_Enable are set when you write to the data set, clear, or toggle register, respectively. The bit pattern on the peripheral bus port is then used to perform the requested function on the port data register.
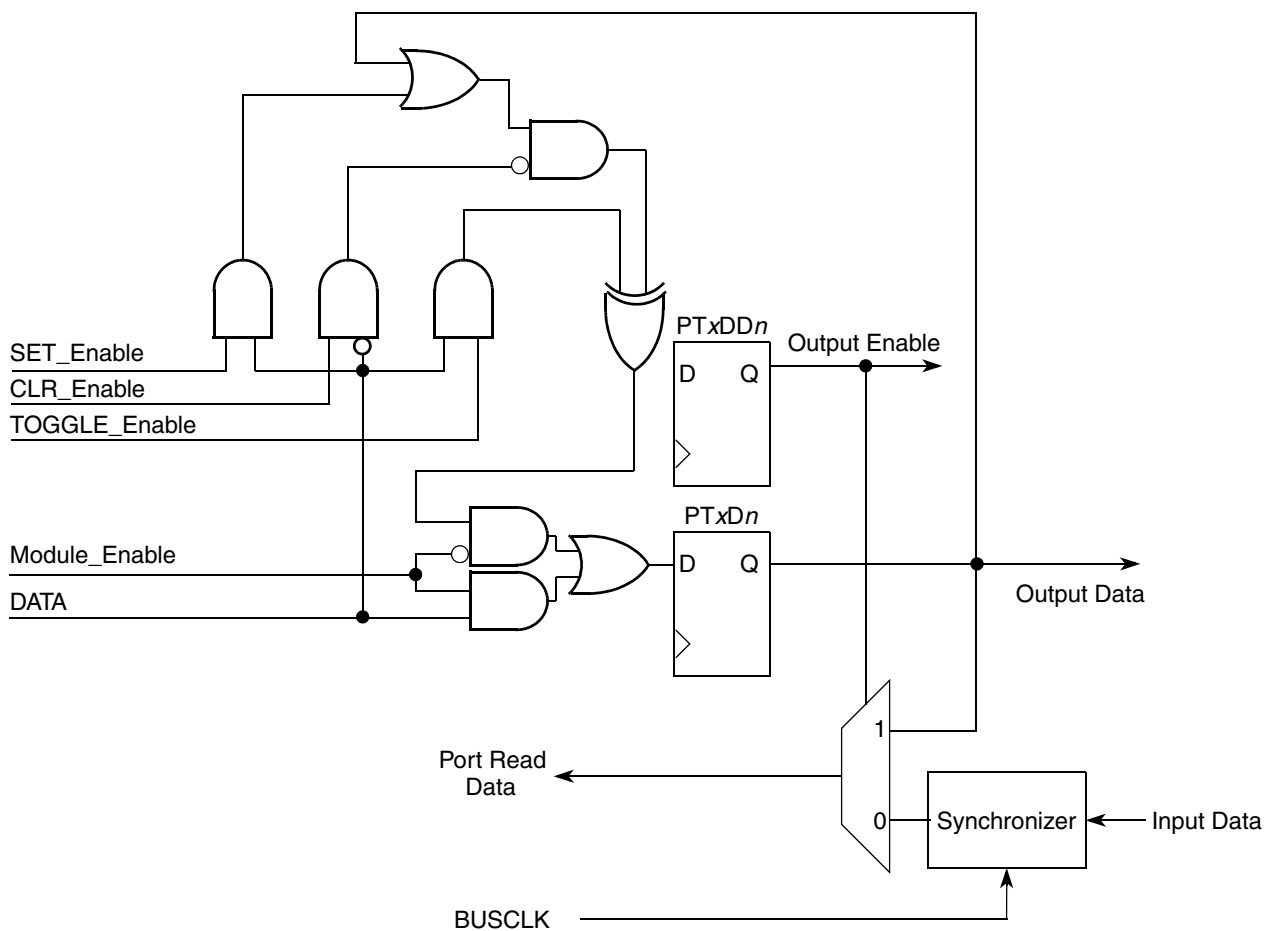
Figure 6-2. Parallel I/O Block Diagram Equipped with SET/CLR Functionality: Ports C & E

MCF51QE128 MCU Series Reference Manual, Rev. 3

The set/clear/toggle functionality allows software to affect an individual bit with a single write instruction, rather than using a read-modify-write sequence.

6.3.1 Port Data Set Registers

The port data set registers (PT*x*SET) are write-only registers associated with ports C and E. Writing to these registers has the result: PortData = PortData || SetPattern. A subsequent port data register (PT*x*D) read reflects the changed result.

6.3.2 Port Data Clear Registers

The port data clear registers (PT*x*CLR) are write-only registers associated with ports C and E. Writing to these registers has the result: PortData = PortData && ~ClearPattern. A subsequent port data register (PT*x*D) read reflects the changed result.

6.3.3 Port Data Toggle Register

The port data toggle registers (PT*x*TOG) are write-only registers associated with ports C & E. Writing to these registers has the result: PortData = PortData ^ TogglePattern. A subsequent port data register (PT*x*D) read reflects the changed result.

6.4 V1 ColdFire Rapid GPIO Functionality

The set/clear/toggle functionality descibed in Section 6.3, "Port Data Set, Clear and Toggle Data Registers," resides on the device peripheral bus. The V1 ColdFire core is capable of performing higher speed I/O via its local bus, which does not have latency penalties associated with the on-chip peripheral bus bridge. The Rapid GPIO module contains data, direction, and enable registers along with set, clear, and toggle registers, which are based at address 0x(00)C0_0000. This functionality can be programmed to take priority on ports C and E.

This functionality is further defined in Chapter 9, "Rapid GPIO (RGPIO)".

6.5 Keyboard Interrupts

Some port A, some port B, and all port D pins can be configured as keyboard interrupt inputs and as an external means of waking the MCU from stop or wait low-power modes. The block diagram for each keyboard interrupt logic is shown Figure 6-3.
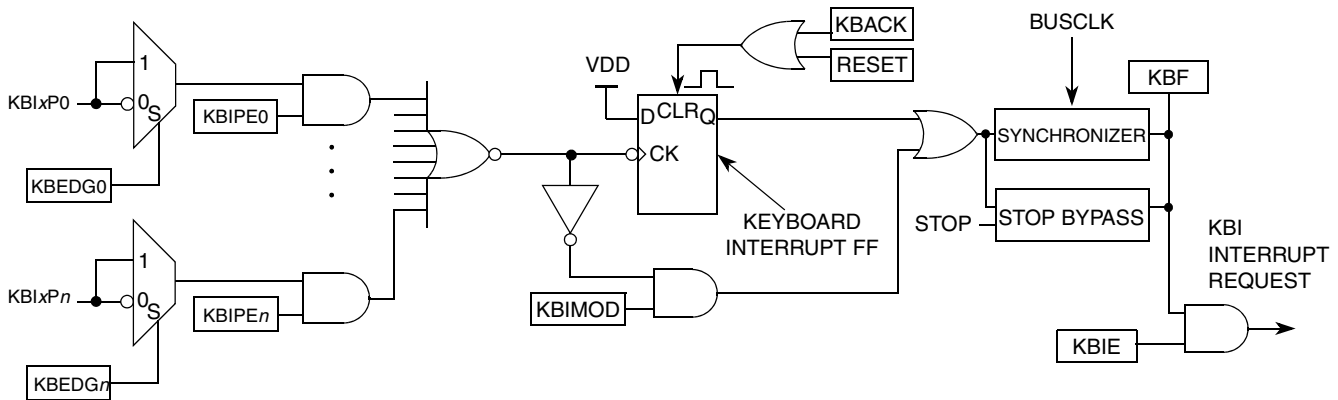
Figure 6-3. Port Interrupt Block Diagram

Writing to the KBIPE*n* bits in the keyboard *x* interrupt pin enable register (KBI*x*PE) independently enables or disables each port pin. Each port can be configured as edge sensitive or edge and level sensitive based on KBI*x*SC[KBIMOD]. Edge sensitivity can be software programmed to be falling or rising; the level can be low or high. The polarity of the edge or edge and level sensitivity is selected using the KBEDG*n* bits in the keyboard interrupt edge select register (KBI*x*ES).

Synchronous logic detects edges. Prior to detecting an edge, enabled port inputs must be at the deasserted logic level. A falling edge is detected when an enabled port input signal is seen as a logic 1 (the deasserted level) during one bus cycle and then a logic 0 (the asserted level) during the next cycle. A rising edge is detected when the input signal is seen as a logic 0 during one bus cycle and then a logic 1 during the next cycle.

6.5.1   Edge Only Sensitivity

A valid edge on an enabled port pin sets KBI*x*SC[KBF]. If the KBI*x*SC[KBIE] bit is set, an interrupt request is generated to the CPU. Clearing of KBF is accomplished by writing a 1 to KBI*x*SC[KBACK].

6.5.2   Edge and Level Sensitivity

A valid edge or level on an enabled port pin sets KBI*x*SC[KBF]. If KBI*x*SC[KBIE] is set, an interrupt request is generated to the CPU. Clearing of KBF is accomplished by writing a 1 to KBI*x*SC[KBACK], provided all enabled port inputs are at their deasserted levels. KBF remains set if any enabled port pin is asserted while attempting to clear by writing a 1 to KBACK.

6.5.3   Pull-up/Pull-down Resistors

The keyboard interrupt pins can be configured to use an internal pull-up/pull-down resistor using the associated I/O port pull-up enable register. If an internal resistor is enabled, the KBI*x*ES register selects whether the resistor is a pull-up (KBEDG*n* = 0) or a pull-down (KBEDG*n* = 1).

6.5.4 Keyboard Interrupt Initialization

When an interrupt pin is first enabled, it is possible to get a false interrupt flag. To prevent a false interrupt request during pin interrupt initialization, do the following:

1. Mask interrupts by clearing KBIxSC[KBIE].
2. Select the pin polarity by setting the appropriate KBIxES[KBEDGn] bits.
3. If using internal pull-up/pull-down device, configure the associated pull enable bits in KBIxPE.
4. Enable the interrupt pins by setting the appropriate KBIxPE[KBIPEn] bits.
5. Write to KBIxSC[KBACK] to clear any false interrupts.
6. Set KBIxSC[KBIE] to enable interrupts.

6.6 Pin Behavior in Stop Modes

Pin behavior following execution of a STOP instruction depends on the entered stop mode. An explanation of pin behavior for the various stop modes follows:

- Stop2 mode is a partial power-down mode, whereby I/O latches are maintained in their state as before the STOP instruction was executed (port states are lost and need to be restored upon exiting stop2). CPU register status and the state of I/O registers should be saved in RAM before the STOP instruction is executed to place the MCU in stop2 mode.
  Upon recovery from stop2 mode, before accessing any I/O, examine the state of the SPMSC2[PPDF] bit. If the PPDF bit is cleared, I/O must be initialized as if a power-on-reset had occurred. If the PPDF bit is set, I/O register states should be restored from the values saved in RAM before the STOP instruction was executed and peripherals may require initialization or restoration to their pre-stop condition. Then, write a 1 to the SPMSC2[PPDACK] bit. Access to I/O is now permitted again in the user application program.
- In stop3 and stop4 modes, all I/O is maintained because internal logic circuity stays powered. Upon recovery, normal I/O function is available to the user.

6.7 Parallel I/O, Keyboard Interrupt, and Pin Control Registers

This section provides information about the registers associated with the parallel I/O ports. The data, data direction registers, and keyboard interrupt registers are located in page zero of the memory map. The pull-up, slew rate, drive strength, and interrupt control registers are located in the high page section of the memory map.

Refer to tables in Chapter 4, "Memory," for the absolute address assignments for all parallel I/O and their pin control registers. This section refers to registers and control bits only by their names. A Freescale Semiconductor-provided equate or header file normally is translates these names into the appropriate absolute addresses.

6.7.1 Port A Registers

Port A is controlled by the registers listed below.

The PTA4 and PTA5 pins are unique. PTA4 is an output only, so the control bits for the input functions do not have any effect on this pin. PTA5, when configured as an output, is open drain. Therefore, the drive strength and slew rate controls have no effect on this pin.

6.7.1.1 Port A Data Register (PTAD)

	7	6	5	4	3	2	1	0
R W	PTAD7	PTAD6	PTAD5	PTAD4[1]	PTAD3	PTAD2	PTAD1	PTAD0
Reset:	0	0	0	0	0	0	0	0
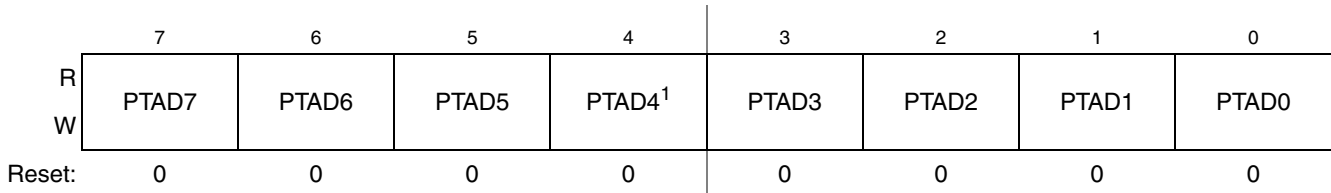
[1] Reads of bit PTAD4 always return the contents of PTAD4, regardless of the value stored in the PTADD4 bit.

Figure 6-4. Port A Data Register (PTAD)

Table 6-1. PTAD Register Field Descriptions

Field	Description
7–0 PTAD*n*	Port A Data Register Bits. For port A pins configured as inputs, reads return the logic level on the pin. For port A pins configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port A pins configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTAD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pull-ups/pull-downs disabled.

6.7.1.2 Port A Data Direction Register (PTADD)

	7	6	5	4	3	2	1	0
R W	PTADD7	PTADD6	PTADD5	PTADD4[1]	PTADD3	PTADD2	PTADD1	PTADD0
Reset:	0	0	0	0	0	0	0	0
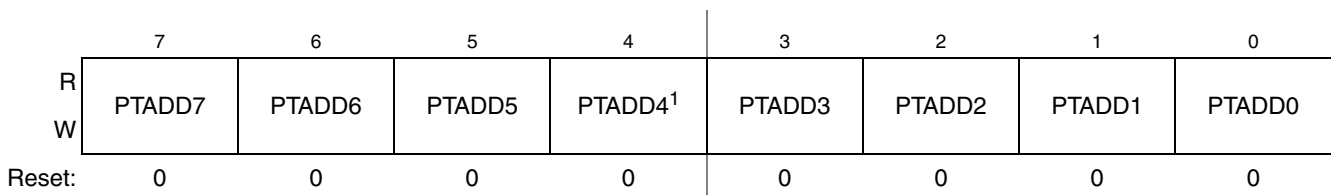
[1] PTADD4 has no effect on the output-only PTA4 pin.

Figure 6-5. Port A Data Direction Register (PTADD)

Table 6-2. PTADD Register Field Descriptions

Field	Description
7–0 PTADD*n*	Data Direction for Port A Bits. These read/write bits control the direction of port A pins and what is read for PTAD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port A bit *n* and PTAD reads return the contents of PTAD*n*.

6.7.1.3 Port A Pull Enable Register (PTAPE)

The port A pull enable register enables pull-ups on the corresponding PTA pin. In some cases, a pull-down device is enabled if pull-downs are supported by an alternate pin function (e.g., KBI).

	7	6	5	4	3	2	1	0
R	PTAPE7	PTAPE6	PTAPE5	PTAPE4[1]	PTAPE3	PTAPE2	PTAPE1	PTAPE0
W								
Reset:	0	0	0	0	0	0	0	0
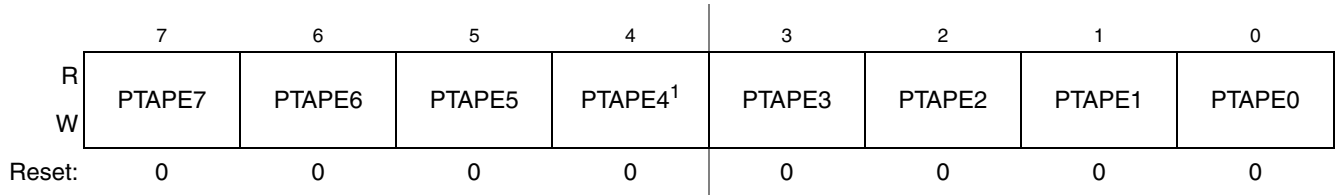
[1] PTAPE4 has no effect on the output-only PTA4 pin.

Figure 6-6. Internal Pull Enable for Port A Register (PTAPE)

Table 6-3. PTAPE Register Field Descriptions

Field	Description
7–0 PTAPE*n*	Internal Pull Enable for Port A Bits. Each of these control bits determines if the internal pull-up or pull-down device is enabled for the associated PTA pin. For port A pins configured as outputs, these bits have no effect and the internal pull devices are disabled. 0 Internal pull-up/pull-down device disabled for port A bit *n*. 1 Internal pull-up/pull-down device enabled for port A bit *n*.

6.7.1.4   Port A Slew Rate Enable Register (PTASE)

	7	6	5	4	3	2	1	0
R	PTASE7	PTASE6	PTASE5[1]	PTASE4	PTASE3	PTASE2	PTASE1	PTASE0
W								
Reset:	0	0	0	0	0	0	0	0

[1] PTASE5 has no effect on the open drain PTA5 pin.

Figure 6-7. Slew Rate Enable for Port A Register (PTASE)

Table 6-4. PTASE Register Field Descriptions

Field	Description
7–0 PTASE*n*	Output Slew Rate Enable for Port A Bits. Each of these control bits determines if the output slew rate control is enabled for the associated PTA pin. For port A pins configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port A bit *n*. 1 Output slew rate control enabled for port A bit *n*.

6.7.1.5   Port A Drive Strength Selection Register (PTADS)

	7	6	5	4	3	2	1	0
R	PTADS7	PTADS6	PTADS5[1]	PTADS4	PTADS3	PTADS2	PTADS1	PTADS0
W								
Reset:	0	0	0	0	0	0	0	0

[1] PTADS5 has no effect on the open drain PTA5 pin.

Figure 6-8. Drive Strength Selection for Port A Register (PTADS)

Table 6-5. PTADS Register Field Descriptions

Field	Description
7–0 PTADS*n*	Output Drive Strength Selection for Port A Bits. Each of these control bits selects between low and high output drive for the associated PTA pin. For port A pins configured as inputs, these bits have no effect. 0 Low output drive strength selected for port A bit *n*. 1 High output drive strength selected for port A bit *n*.

6.7.2   Port B Registers

Port B is controlled by the registers listed below.

6.7.2.1   Port B Data Register (PTBD)

	7	6	5	4	3	2	1	0
R W	PTBD7	PTBD6	PTBD5	PTBD4	PTBD3	PTBD2	PTBD1	PTBD0
Reset:	0	0	0	0	0	0	0	0
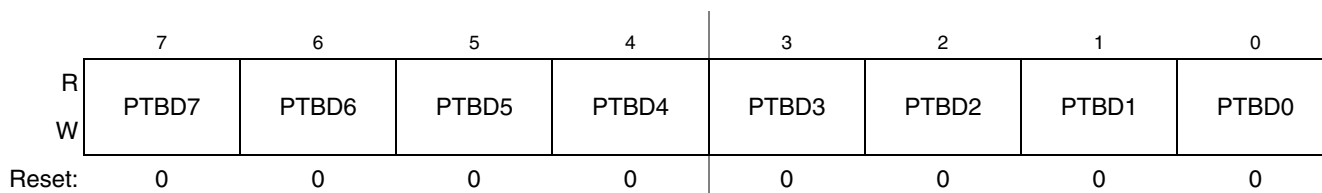
Figure 6-9. Port B Data Register (PTBD)

Table 6-6. PTBD Register Field Descriptions

Field	Description
7–0 PTBD*n*	Port B Data Register Bits. For port B pins configured as inputs, reads return the logic level on the pin. For port B pins configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port B pins configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTBD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pull-ups/pull-downs disabled.

6.7.2.2   Port B Data Direction Register (PTBDD)

	7	6	5	4	3	2	1	0
R W	PTBDD7	PTBDD6	PTBDD5	PTBDD4	PTBDD3	PTBDD2	PTBDD1	PTBDD0
Reset:	0	0	0	0	0	0	0	0
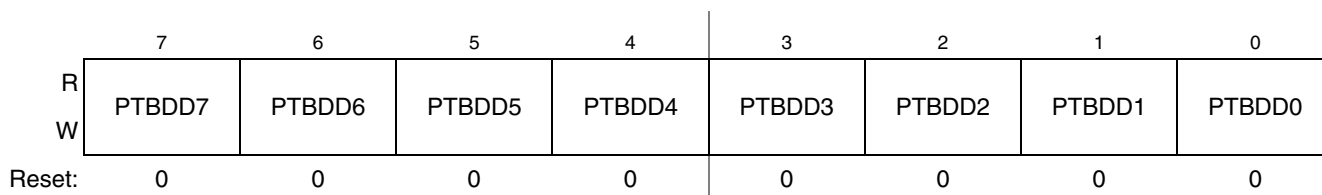
Figure 6-10. Port B Data Direction Register (PTBDD)

Table 6-7. PTBDD Register Field Descriptions

Field	Description
7–0 PTBDD*n*	Data Direction for Port B Bits. These read/write bits control the direction of port B pins and what is read for PTBD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port B bit *n* and PTBD reads return the contents of PTBD*n*.

6.7.2.3   Port B Pull Enable Register (PTBPE)

The port B pull enable register enables pull-ups on the corresponding PTB pin. In some cases, a pull-down device is enabled if pull-downs are supported by an alternate pin function (e.g., KBI).

	7	6	5	4	3	2	1	0
R W	PTBPE7	PTBPE6	PTBPE5	PTBPE4	PTBPE3	PTBPE2	PTBPE1	PTBPE0
Reset:	0	0	0	0	0	0	0	0
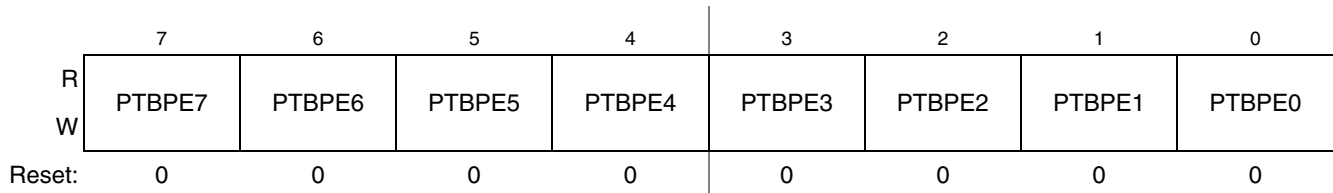
Figure 6-11. Internal Pull Enable for Port B Register (PTBPE)

Table 6-8. PTBPE Register Field Descriptions

Field	Description
7–0 PTBPE*n*	Internal Pull Enable for Port B Bits. Each of these control bits determines if the internal pull-up or pull-down device is enabled for the associated PTB pin. For port B pins configured as outputs, these bits have no effect and the internal pull devices are disabled. 0 Internal pull-up/pull-down device disabled for port B bit *n*. 1 Internal pull-up/pull-down device enabled for port B bit *n*.

6.7.2.4   Port B Slew Rate Enable Register (PTBSE)

	7	6	5	4	3	2	1	0
R W	PTBSE7	PTBSE6	PTBSE5	PTBSE4	PTBSE3	PTBSE2	PTBSE1	PTBSE0
Reset:	0	0	0	0	0	0	0	0
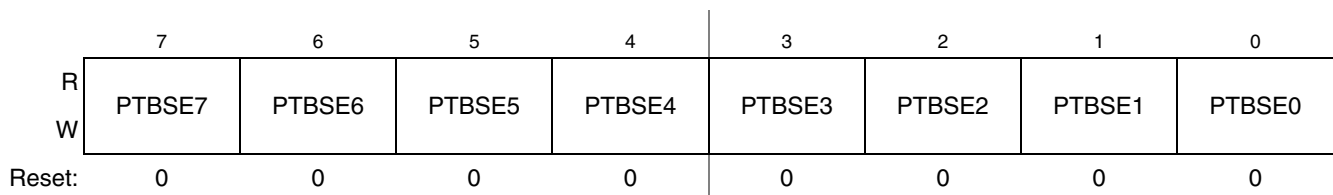
Figure 6-12. Slew Rate Enable for Port B Register (PTBSE)

Table 6-9. PTBSE Register Field Descriptions

Field	Description
7–0 PTBSE*n*	Output Slew Rate Enable for Port B Bits. Each of these control bits determines if the output slew rate control is enabled for the associated PTB pin. For port B pins configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port B bit *n*. 1 Output slew rate control enabled for port B bit *n*.

6.7.2.5 Port B Drive Strength Selection Register (PTBDS)

	7	6	5	4	3	2	1	0
R W	PTBDS7	PTBDS6	PTBDS5	PTBDS4	PTBDS3	PTBDS2	PTBDS1	PTBDS0
Reset:	0	0	0	0	0	0	0	0
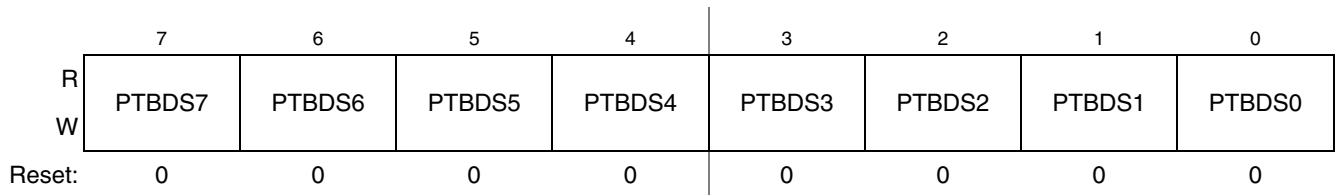
Figure 6-13. Drive Strength Selection for Port B Register (PTBDS)

Table 6-10. PTBDS Register Field Descriptions

Field	Description
7–0 PTBDS*n*	Output Drive Strength Selection for Port B Bits. Each of these control bits selects between low and high output drive for the associated PTB pin. For port B pins configured as inputs, these bits have no effect. 0 Low output drive strength selected for port B bit *n*. 1 High output drive strength selected for port B bit *n*.

6.7.3 Port C Registers

Port C is controlled by the registers listed below.

6.7.3.1 Port C Data Register (PTCD)

	7	6	5	4	3	2	1	0
R W	PTCD7	PTCD6	PTCD5	PTCD4	PTCD3	PTCD2	PTCD1	PTCD0
Reset:	0	0	0	0	0	0	0	0
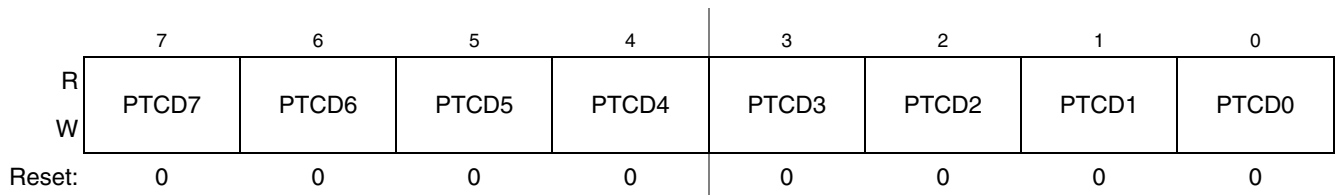
Figure 6-14. Port C Data Register (PTCD)

Table 6-11. PTCD Register Field Descriptions

Field	Description
7–0 PTCD*n*	Port C Data Register Bits. For port C pins configured as inputs, reads return the logic level on the pin. For port C pins configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port C pins configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTCD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pull-ups disabled.

6.7.3.2 Port C Data Direction Register (PTCDD)

	7	6	5	4	3	2	1	0
R	PTCDD7	PTCDD6	PTCDD5	PTCDD4	PTCDD3	PTCDD2	PTCDD1	PTCDD0
W								
Reset:	0	0	0	0	0	0	0	0
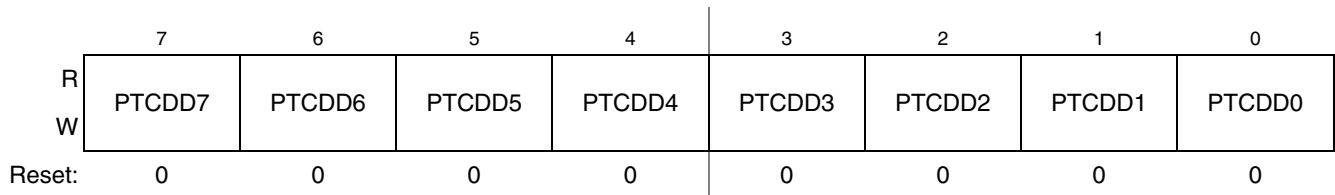
Figure 6-15. Port C Data Direction Register (PTCDD)

Table 6-12. PTCDD Register Field Descriptions

Field	Description
7–0 PTCDD*n*	Data Direction for Port C Bits. These read/write bits control the direction of port C pins and what is read for PTCD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port C bit *n* and PTCD reads return the contents of PTCD*n*.

6.7.3.3 Port C Data Set Register (PTCSET)

	7	6	5	4	3	2	1	0
R								
W	PTCSET7	PTCSET6	PTCSET5	PTCSET4	PTCSET3	PTCSET2	PTCSET1	PTCSET0
Reset:	0	0	0	0	0	0	0	0
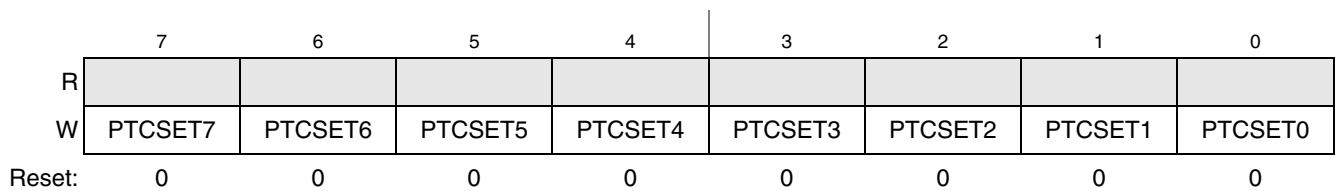
Figure 6-16. Port C Data Set Register (PTCSET)

Table 6-13. PTCSET Register Field Descriptions

Field	Description
7–0 PTCSET*n*	Data Set for Port C Bits. Writing 1 to any bit in this location sets the corresponding bit in the data register. Writing a zero to any bit in this register has no effect. 0 Corresponding PTCD*n* maintains current value. 1 Corresponding PTCD*n* is set.

6.7.3.4 Port C Data Clear Register (PTCCLR)

	7	6	5	4	3	2	1	0
R								
W	PTCCLR7	PTCCLR6	PTCCLR5	PTCCLR4	PTCCLR3	PTCCLR2	PTCCLR1	PTCCLR0
Reset:	0	0	0	0	0	0	0	0
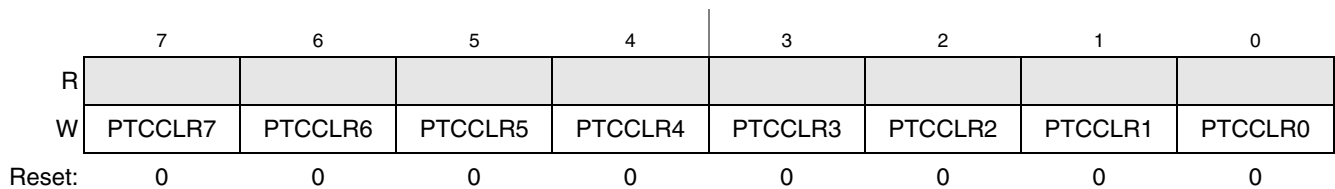
Figure 6-17. Port C Data Clear Register (PTCCLR)

Table 6-14. PTCCLR Register Field Descriptions

Field	Description
7–0 PTCCLR*n*	Data Clear for Port C Bits. Writing 0 to any bit in this location clears the corresponding bit in the data register. Writing a one to any bit in this register has no effect. 0 Corresponding PTCD*n* maintains current value. 1 Corresponding PTCD*n* is cleared.

6.7.3.5   Port C Toggle Register (PTCTOG)

	7	6	5	4	3	2	1	0
R								
W	PTCTOG7	PTCTOG6	PTCTOG5	PTCTOG4	PTCTOG3	PTCTOG2	PTCTOG1	PTCTOG0
Reset:	0	0	0	0	0	0	0	0

Figure 6-18. Port C Toggle Enable Register (PTCTOG)

Table 6-15. PTCTOG Register Field Descriptions

Field	Description
7–0 PTCTOG*n*	Toggle Enable for Port C Bits. Writing 1 to any bit in this location toggles the corresponding bit in the data register. Writing a zero to any bit in this register has no effect. 0 Corresponding PTCD*n* maintains current value. 1 Corresponding PTCD*n* is toggled once.

6.7.3.6   Port C Pull Enable Register (PTCPE)

The port C pull enable register enables pull-ups on the corresponding PTC pin. In some cases, a pull-down device is enabled if pull-downs are supported by an alternate pin function (e.g., KBI).

	7	6	5	4	3	2	1	0
R W	PTCPE7	PTCPE6	PTCPE5	PTCPE4	PTCPE3	PTCPE2	PTCPE1	PTCPE0
Reset:	0	0	0	0	0	0	0	0

Figure 6-19. Internal Pull Enable for Port C Register (PTCPE)

Table 6-16. PTCPE Register Field Descriptions

Field	Description
7–0 PTCPE*n*	Internal Pull Enable for Port C Bits. Each of these control bits determines if the internal pull-up device is enabled for the associated PTC pin. For port C pins configured as outputs, these bits have no effect and the internal pull devices are disabled. 0 Internal pull-up device disabled for port C bit *n*. 1 Internal pull-up device enabled for port C bit *n*.

6.7.3.7 Port C Slew Rate Enable Register (PTCSE)

	7	6	5	4	3	2	1	0
R	PTCSE7	PTCSE6	PTCSE5	PTCSE4	PTCSE3	PTCSE2	PTCSE1	PTCSE0
W								
Reset:	0	0	0	0	0	0	0	0
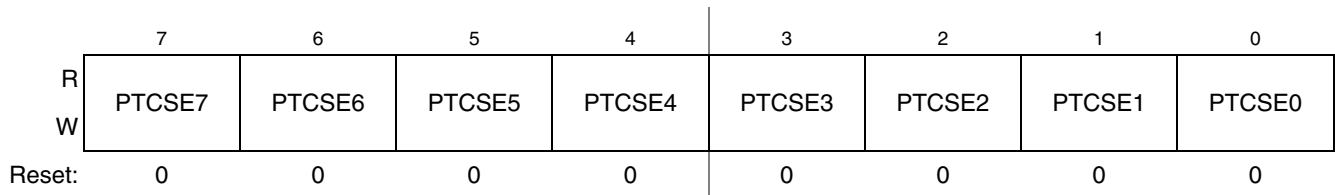
Figure 6-20. Slew Rate Enable for Port C Register (PTCSE)

Table 6-17. PTCSE Register Field Descriptions

Field	Description
7–0 PTCSE*n*	Output Slew Rate Enable for Port C Bits. Each of these control bits determines if the output slew rate control is enabled for the associated PTC pin. For port C pins configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port C bit *n*. 1 Output slew rate control enabled for port C bit *n*.

6.7.3.8 Port C Drive Strength Selection Register (PTCDS)

	7	6	5	4	3	2	1	0
R	PTCDS7	PTCDS6	PTCDS5	PTCDS4	PTCDS3	PTCDS2	PTCDS1	PTCDS0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 6-21. Drive Strength Selection for Port C Register (PTCDS)

Table 6-18. PTCDS Register Field Descriptions

Field	Description
7–0 PTCDS*n*	Output Drive Strength Selection for Port C Bits. Each of these control bits selects between low and high output drive for the associated PTC pin. For port C pins configured as inputs, these bits have no effect. 0 Low output drive strength selected for port C bit *n*. 1 High output drive strength selected for port C bit *n*.

6.7.4 Port D Registers

Port D is controlled by the registers listed below.

6.7.4.1 Port D Data Register (PTDD)

	7	6	5	4	3	2	1	0
R	PTDD7	PTDD6	PTDD5	PTDD4	PTDD3	PTDD2	PTDD1	PTDD0
W								
Reset:	0	0	0	0	0	0	0	0
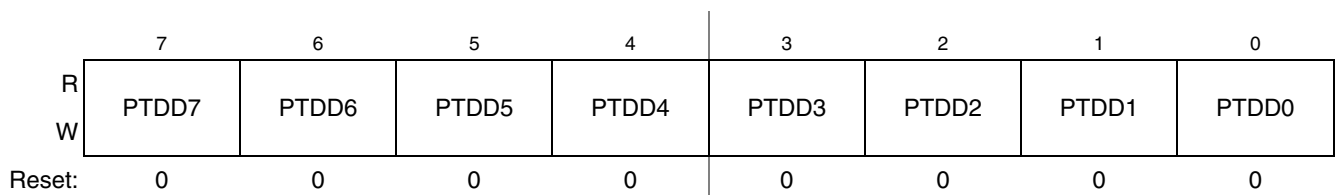
Figure 6-22. Port D Data Register (PTDD)

Table 6-19. PTDD Register Field Descriptions

Field	Description
7–0 PTDD*n*	Port D Data Register Bits. For port D pins configured as inputs, reads return the logic level on the pin. For port D pins configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port D pins configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTDD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pull-ups/pull-downs disabled.

6.7.4.2 Port D Data Direction Register (PTDDD)

	7	6	5	4	3	2	1	0
R W	PTDDD7	PTDDD6	PTDDD5	PTDDD4	PTDDD3	PTDDD2	PTDDD1	PTDDD0
Reset:	0	0	0	0	0	0	0	0
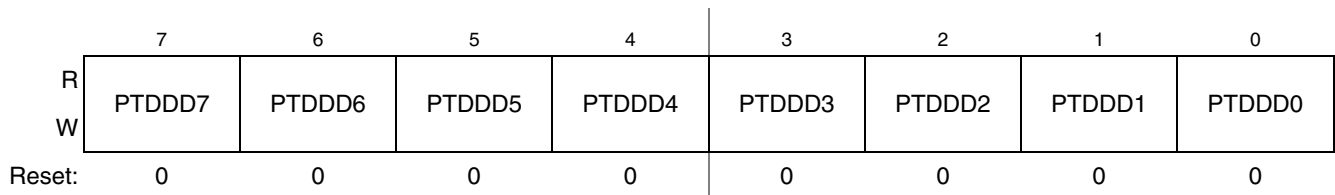
Figure 6-23. Port D Data Direction Register (PTDDD)

Table 6-20. PTDDD Register Field Descriptions

Field	Description
7–0 PTDDD*n*	Data Direction for Port D Bits. These read/write bits control the direction of port D pins and what is read for PTDD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port D bit *n* and PTDD reads return the contents of PTDD*n*.

6.7.4.3 Port D Pull Enable Register (PTDPE)

The port D pull enable register enables pull-ups on the corresponding PTD pin. In some cases, a pull-down device is enabled if pull-downs are supported by an alternate pin function (e.g., KBI).
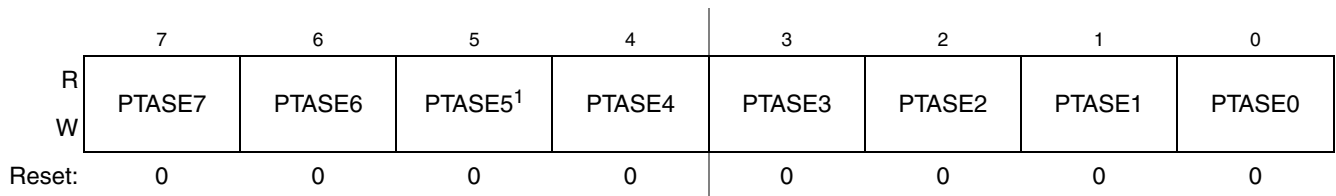
	7	6	5	4	3	2	1	0
R W	PTDPE7	PTDPE6	PTDPE5	PTDPE4	PTDPE3	PTDPE2	PTDPE1	PTDPE0
Reset:	0	0	0	0	0	0	0	0

Figure 6-24. Internal Pull Enable for Port D Register (PTDPE)

Table 6-21. PTDPE Register Field Descriptions

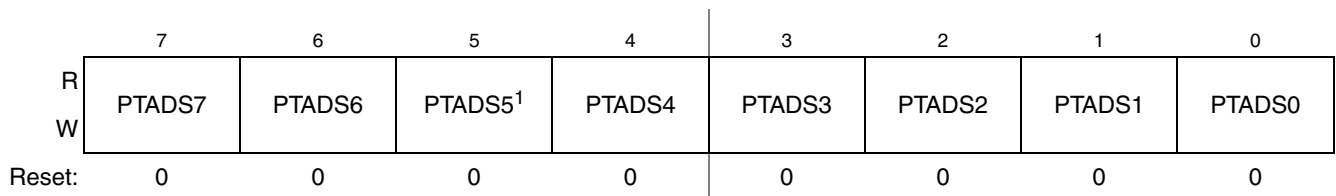| Field	Description
7–0 PTDPE*n*	Internal Pull Enable for Port D Bits. Each of these control bits determines if the internal pull-up or pull-down device is enabled for the associated PTD pin. For port D pins configured as outputs, these bits have no effect and the internal pull devices are disabled. 0 Internal pull-up/pull-down device disabled for port D bit *n*. 1 Internal pull-up/pull-down device enabled for port D bit *n*.

6.7.4.4 Port D Slew Rate Enable Register (PTDSE)

	7	6	5	4	3	2	1	0
R	PTDSE7	PTDSE6	PTDSE5	PTDSE4	PTDSE3	PTDSE2	PTDSE1	PTDSE0
W								
Reset:	0	0	0	0	0	0	0	0
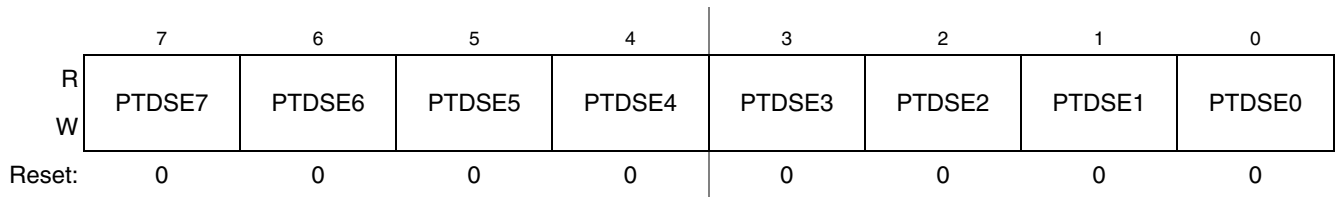
Figure 6-25. Slew Rate Enable for Port D Register (PTDSE)

Table 6-22. PTDSE Register Field Descriptions

Field	Description
7–0 PTDSE*n*	Output Slew Rate Enable for Port D Bits. Each of these control bits determines if the output slew rate control is enabled for the associated PTD pin. For port D pins configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port D bit *n*. 1 Output slew rate control enabled for port D bit *n*.

6.7.4.5 Port D Drive Strength Selection Register (PTDDS)

	7	6	5	4	3	2	1	0
R	PTDDS7	PTDDS6	PTDDS5	PTDDS4	PTDDS3	PTDDS2	PTDDS1	PTDDS0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 6-26. Drive Strength Selection for Port D Register (PTDDS)

Table 6-23. PTDDS Register Field Descriptions

Field	Description
7–0 PTDDS*n*	Output Drive Strength Selection for Port D Bits. Each of these control bits selects between low and high output drive for the associated PTD pin. For port D pins configured as inputs, these bits have no effect. 0 Low output drive strength selected for port D bit *n*. 1 High output drive strength selected for port D bit *n*.

6.7.5 Port E Registers

Port E is controlled by the registers listed below.

6.7.5.1 Port E Data Register (PTED)

	7	6	5	4	3	2	1	0
R	PTED7	PTED6	PTED5	PTED4	PTED3	PTED2	PTED1	PTED0
W								
Reset:	0	0	0	0	0	0	0	0
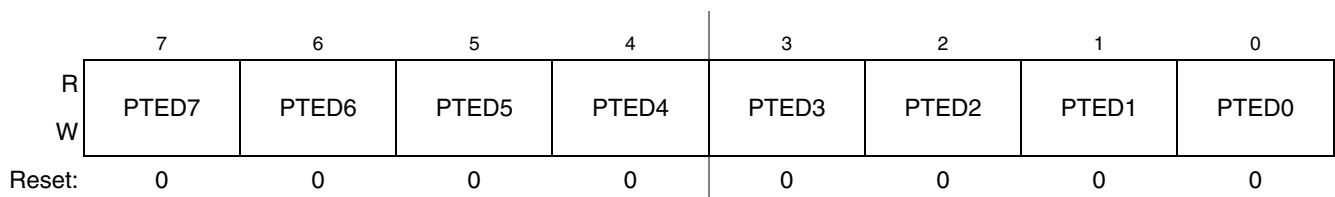
Figure 6-27. Port E Data Register (PTED)

Table 6-24. PTED Register Field Descriptions

Field	Description
7–0 PTED*n*	Port E Data Register Bits. For port E pins configured as inputs, reads return the logic level on the pin. For port E pins configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port E pins configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTED to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pull-ups disabled.

6.7.5.2   Port E Data Direction Register (PTEDD)

	7	6	5	4	3	2	1	0
R W	PTEDD7	PTEDD6	PTEDD5	PTEDD4	PTEDD3	PTEDD2	PTEDD1	PTEDD0
Reset:	0	0	0	0	0	0	0	0
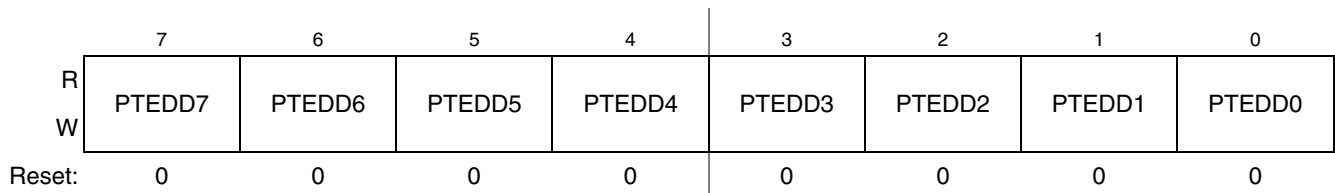
Figure 6-28. Port E Data Direction Register (PTEDD)

Table 6-25. PTEDD Register Field Descriptions

Field	Description
7–0 PTEDD*n*	Data Direction for Port E Bits. These read/write bits control the direction of port E pins and what is read for PTED reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port E bit *n* and PTED reads return the contents of PTED*n*.

6.7.5.3   Port E Data Set Register (PTESET)

	7	6	5	4	3	2	1	0
R								
W	PTESET7	PTESET6	PTESET5	PTESET4	PTESET3	PTESET2	PTESET1	PTESET0
Reset:	0	0	0	0	0	0	0	0
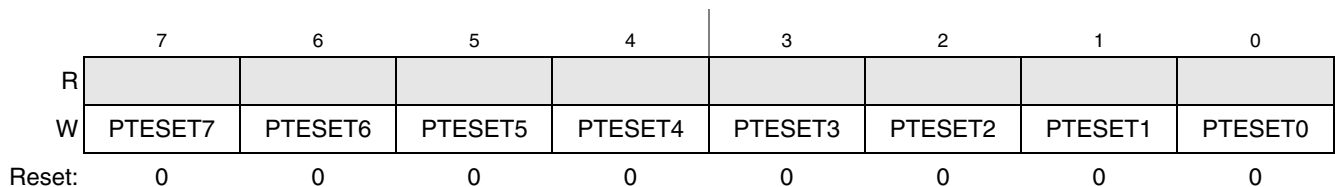
Figure 6-29. Port E Data Set Register (PTESET)

Table 6-26. PTESET Register Field Descriptions

Field	Description
7–0 PTESET*n*	Data Set for Port E Bits. Writing 1 to any bit in this location sets the corresponding bit in the data register. Writing a zero to any bit in this register has no effect. 0 Corresponding PTED*n* maintains current value. 1 Corresponding PTED*n* is set.

6.7.5.4 Port E Data Clear Register (PTECLR)

	7	6	5	4	3	2	1	0
R								
W	PTECLR7	PTECLR6	PTECLR5	PTECLR4	PTECLR3	PTECLR2	PTECLR1	PTECLR0
Reset:	0	0	0	0	0	0	0	0
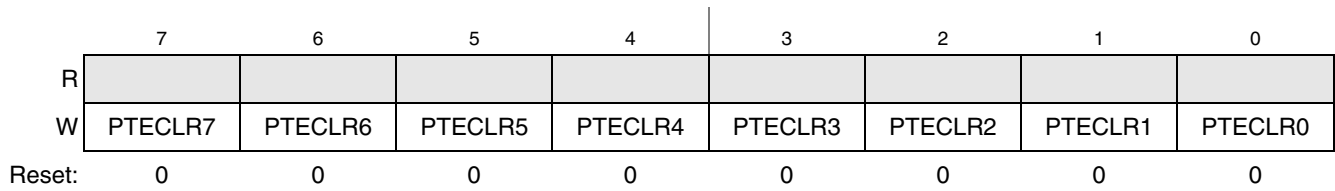
Figure 6-30. Port E Data Clear Register (PTECLR)

Table 6-27. PTECLR Register Field Descriptions

Field	Description
7–0 PTECLR*n*	Data Clear for Port E Bits. Writing 0 to any bit in this location clears the corresponding bit in the data register. Writing a one to any bit in this register has no effect. 0 Corresponding PTED*n* maintains current value. 1 Corresponding PTED*n* is cleared.

6.7.5.5 Port E Toggle Register (PTETOG)

	7	6	5	4	3	2	1	0
R								
W	PTETOG7	PTETOG6	PTETOG5	PTETOG4	PTETOG3	PTETOG2	PTETOG1	PTETOG0
Reset:	0	0	0	0	0	0	0	0
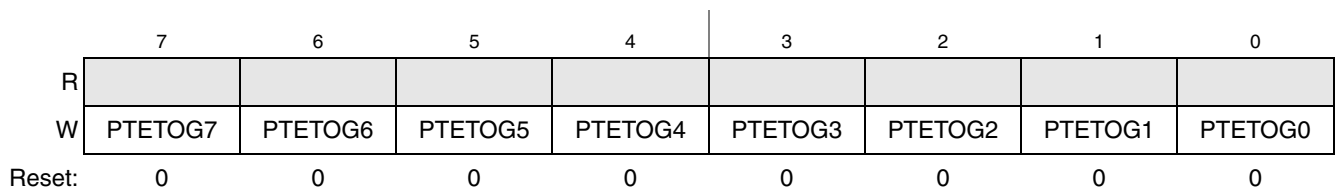
Figure 6-31. Port E Toggle Enable Register (PTETOG)

Table 6-28. PTETOG Register Field Descriptions

Field	Description
7–0 PTETOG*n*	Toggle Enable for Port E Bits. Writing 1 to any bit in this location toggles the corresponding bit in the data register. Writing a zero to any bit in this register has no effect. 0 Corresponding PTED*n* maintains current value. 1 Corresponding PTED*n* is toggled once.

6.7.5.6 Port E Pull Enable Register (PTEPE)

The port E pull enable register enables pull-ups on the corresponding PTE pin. In some cases, a pull-down device is enabled if pull-downs are supported by an alternate pin function (e.g., KBI).

	7	6	5	4	3	2	1	0
R W	PTEPE7	PTEPE6	PTEPE5	PTEPE4	PTEPE3	PTEPE2	PTEPE1	PTEPE0
Reset:	0	0	0	0	0	0	0	0

Figure 6-32. Internal Pull Enable for Port E Register (PTEPE)

Table 6-29. PTEPE Register Field Descriptions

Field	Description
7–0 PTEPE*n*	Internal Pull Enable for Port E Bits. Each of these control bits determines if the internal pull-up device is enabled for the associated PTE pin. For port E pins configured as outputs, these bits have no effect and the internal pull devices are disabled. 0 Internal pull-up device disabled for port E bit *n*. 1 Internal pull-up device enabled for port E bit *n*.

6.7.5.7 Port E Slew Rate Enable Register (PTESE)

	7	6	5	4	3	2	1	0
R W	PTESE7	PTESE6	PTESE5	PTESE4	PTESE3	PTESE2	PTESE1	PTESE0
Reset:	0	0	0	0	0	0	0	0

Figure 6-33. Slew Rate Enable for Port E Register (PTESE)

Table 6-30. PTESE Register Field Descriptions

Field	Description
7–0 PTESE*n*	Output Slew Rate Enable for Port E Bits. Each of these control bits determines if the output slew rate control is enabled for the associated PTE pin. For port E pins configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port E bit *n*. 1 Output slew rate control enabled for port E bit *n*.

6.7.5.8 Port E Drive Strength Selection Register (PTEDS)

	7	6	5	4	3	2	1	0
R W	PTEDS7	PTEDS6	PTEDS5	PTEDS4	PTEDS3	PTEDS2	PTEDS1	PTEDS0
Reset:	0	0	0	0	0	0	0	0

Figure 6-34. Drive Strength Selection for Port E Register (PTEDS)

Table 6-31. PTEDS Register Field Descriptions

Field	Description
7–0 PTEDS*n*	Output Drive Strength Selection for Port E Bits. Each of these control bits selects between low and high output drive for the associated PTE pin. For port E pins configured as inputs, these bits have no effect. 0 Low output drive strength selected for port E bit *n*. 1 High output drive strength selected for port E bit *n*.

6.7.6 Port F Registers

Port F is controlled by the registers listed below.

6.7.6.1 Port F Data Register (PTFD)

	7	6	5	4	3	2	1	0
R	PTFD7	PTFD6	PTFD5	PTFD4	PTFD3	PTFD2	PTFD1	PTFD0
W								
Reset:	0	0	0	0	0	0	0	0
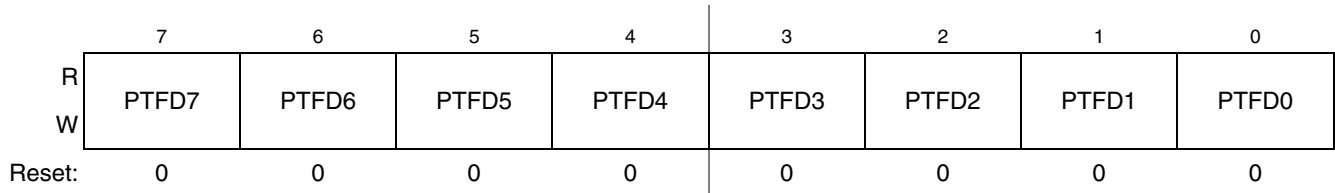
Figure 6-35. Port F Data Register (PTFD)

Table 6-32. PTFD Register Field Descriptions

Field	Description
7–0 PTFD*n*	Port F Data Register Bits. For port F pins configured as inputs, reads return the logic level on the pin. For port F pins configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port F pins configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTFD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pull-ups disabled.

6.7.6.2 Port F Data Direction Register (PTFDD)

	7	6	5	4	3	2	1	0
R	PTFDD7	PTFDD6	PTFDD5	PTFDD4	PTFDD3	PTFDD2	PTFDD1	PTFDD0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 6-36. Port F Data Direction Register (PTFDD)

Table 6-33. PTFDD Register Field Descriptions

Field	Description
7–0 PTFDD*n*	Data Direction for Port F Bits. These read/write bits control the direction of port F pins and what is read for PTFD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port F bit *n* and PTFD reads return the contents of PTFD*n*.

6.7.6.3 Port F Pull Enable Register (PTFPE)

The port F pull enable register enables pull-ups on the corresponding PTF pin. In some cases, a pull-down device is enabled if pull-downs are supported by an alternate pin function (e.g., KBI).

	7	6	5	4	3	2	1	0
R	PTFPE7	PTFPE6	PTFPE5	PTFPE4	PTFPE3	PTFPE2	PTFPE1	PTFPE0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 6-37. Internal Pull Enable for Port F Register (PTFPE)

Table 6-34. PTFPE Register Field Descriptions

Field	Description
7–0 PTFPE*n*	Internal Pull Enable for Port F Bits. Each of these control bits determines if the internal pull-up device is enabled for the associated PTF pin. For port F pins configured as outputs, these bits have no effect and the internal pull devices are disabled. 0 Internal pull-up device disabled for port F bit *n*. 1 Internal pull-up device enabled for port F bit *n*.

6.7.6.4 Port F Slew Rate Enable Register (PTFSE)

	7	6	5	4	3	2	1	0
R	PTFSE7	PTFSE6	PTFSE5	PTFSE4	PTFSE3	PTFSE2	PTFSE1	PTFSE0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 6-38. Slew Rate Enable for Port F Register (PTFSE)

Table 6-35. PTFSE Register Field Descriptions

Field	Description
7–0 PTFSE*n*	Output Slew Rate Enable for Port F Bits. Each of these control bits determines if the output slew rate control is enabled for the associated PTF pin. For port F pins configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port F bit *n*. 1 Output slew rate control enabled for port F bit *n*.

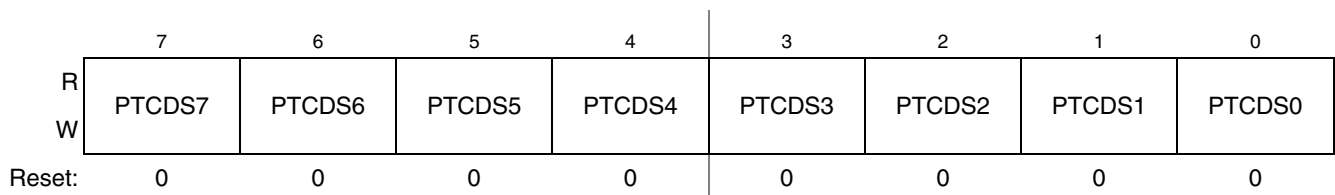### 6.7.6.5 Port F Drive Strength Selection Register (PTFDS)

	7	6	5	4	3	2	1	0
R	PTFDS7	PTFDS6	PTFDS5	PTFDS4	PTFDS3	PTFDS2	PTFDS1	PTFDS0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 6-39. Drive Strength Selection for Port F Register (PTFDS)

Table 6-36. PTFDS Register Field Descriptions

Field	Description
7–0 PTFDS*n*	Output Drive Strength Selection for Port F Bits. Each of these control bits selects between low and high output drive for the associated PTF pin. For port F pins configured as inputs, these bits have no effect. 0 Low output drive strength selected for port F bit *n*. 1 High output drive strength selected for port F bit *n*.

6.7.7 Port G Registers

Port G is controlled by the registers listed below.

6.7.7.1 Port G Data Register (PTGD)

	7	6	5	4	3	2	1	0
R W	PTGD7	PTGD6	PTGD5	PTGD4	PTGD3	PTGD2	PTGD1	PTGD0
Reset:	0	0	0	0	0	0	0	0
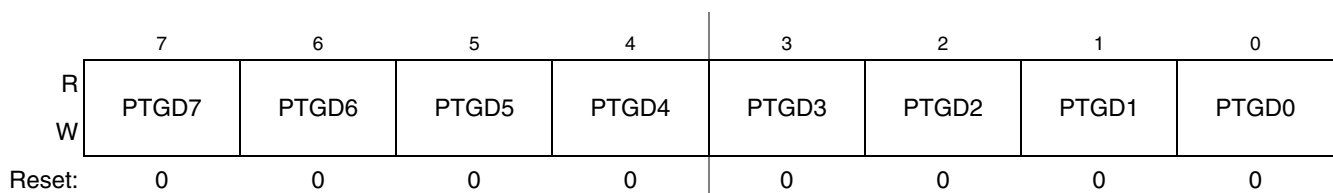
Figure 6-40. Port G Data Register (PTGD)

Table 6-37. PTGD Register Field Descriptions

Field	Description
7–0 PTGD*n*	Port G Data Register Bits. For port G pins configured as inputs, reads return the logic level on the pin. For port G pins configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port G pins configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTGD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pull-ups disabled.

6.7.7.2 Port G Data Direction Register (PTGDD)

	7	6	5	4	3	2	1	0
R W	PTGDD7	PTGDD6	PTGDD5	PTGDD4	PTGDD3	PTGDD2	PTGDD1	PTGDD0
Reset:	0	0	0	0	0	0	0	0

Figure 6-41. Port G Data Direction Register (PTGDD)

Table 6-38. PTGDD Register Field Descriptions

Field	Description
7–0 PTGDD*n*	Data Direction for Port G Bits. These read/write bits control the direction of port G pins and what is read for PTGD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port G bit *n* and PTGD reads return the contents of PTGD*n*.

6.7.7.3 Port G Pull Enable Register (PTGPE)

The port G pull enable register enables pull-ups on the corresponding PTG pin. In some cases, a pull-down device is enabled if pull-downs are supported by an alternate pin function (e.g., KBI).
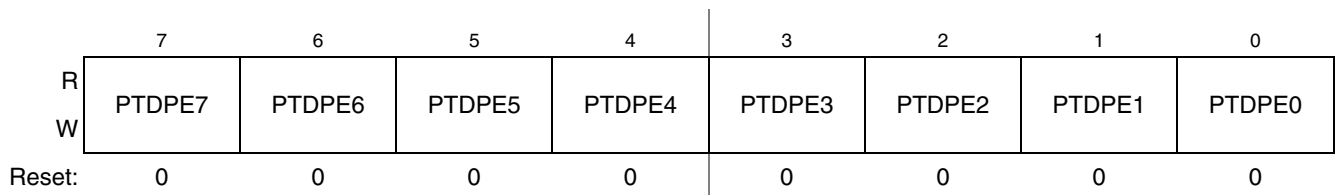
	7	6	5	4	3	2	1	0
R	PTGPE7	PTGPE6	PTGPE5	PTGPE4	PTGPE3	PTGPE2	PTGPE1	PTGPE0
W								
Reset:	0	0	0	0	0	0	0	0
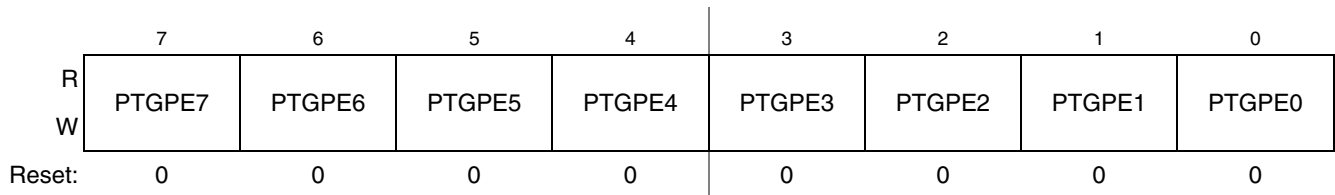
Figure 6-42. Internal Pull Enable for Port G Register (PTGPE)

Table 6-39. PTGPE Register Field Descriptions

Field	Description
7–0 PTGPE*n*	Internal Pull Enable for Port G Bits. Each of these control bits determines if the internal pull-up device is enabled for the associated PTG pin. For port G pins configured as outputs, these bits have no effect and the internal pull devices are disabled. 0 Internal pull-up device disabled for port G bit *n*. 1 Internal pull-up device enabled for port G bit *n*.

6.7.7.4 Port G Slew Rate Enable Register (PTGSE)

	7	6	5	4	3	2	1	0
R	PTGSE7	PTGSE6	PTGSE5	PTGSE4	PTGSE3	PTGSE2	PTGSE1	PTGSE0
W								
Reset:	0	0	0	0	0	0	0	0
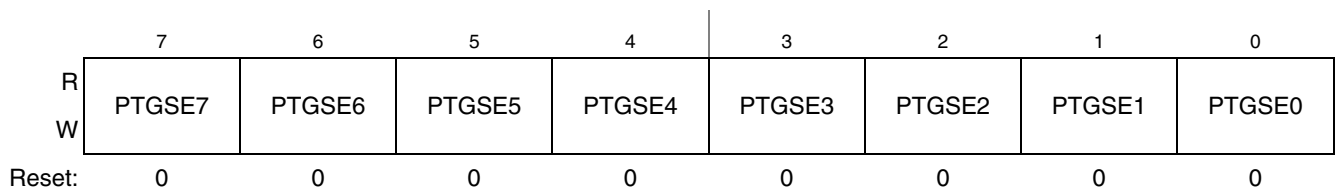
Figure 6-43. Slew Rate Enable for Port G Register (PTGSE)

Table 6-40. PTGSE Register Field Descriptions

Field	Description
7–0 PTGSE*n*	Output Slew Rate Enable for Port G Bits. Each of these control bits determines if the output slew rate control is enabled for the associated PTG pin. For port G pins configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port G bit *n*. 1 Output slew rate control enabled for port G bit *n*.

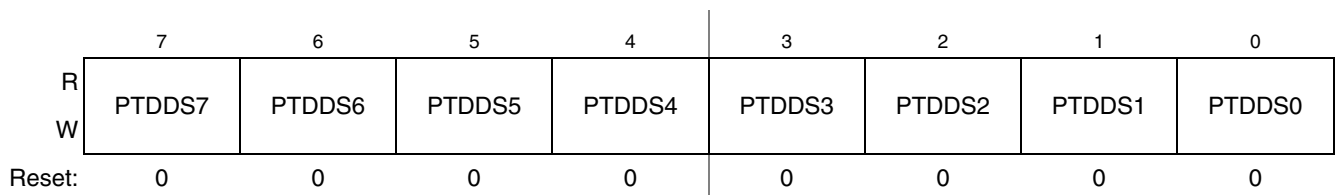### 6.7.7.5 Port G Drive Strength Selection Register (PTGDS)

	7	6	5	4	3	2	1	0
R	PTGDS7	PTGDS6	PTGDS5	PTGDS4	PTGDS3	PTGDS2	PTGDS1	PTGDS0
W								
Reset:	0	0	0	0	0	0	0	0
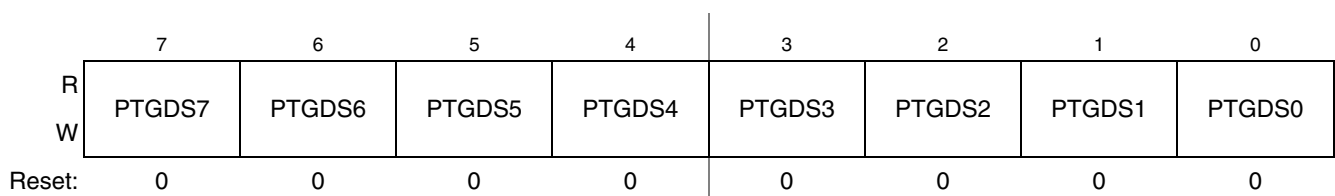
Figure 6-44. Drive Strength Selection for Port G Register (PTGDS)

Table 6-41. PTGDS Register Field Descriptions

Field	Description
7–0 PTGDS*n*	Output Drive Strength Selection for Port G Bits. Each of these control bits selects between low and high output drive for the associated PTG pin. For port G pins configured as inputs, these bits have no effect. 0 Low output drive strength selected for port G bit *n*. 1 High output drive strength selected for port G bit *n*.

6.7.8   Port H Registers

Port H is controlled by the registers listed below.

6.7.8.1   Port H Data Register (PTHD)

	7	6	5	4	3	2	1	0
R W	PTHD7	PTHD6	PTHD5	PTHD4	PTHD3	PTHD2	PTHD1	PTHD0
Reset:	0	0	0	0	0	0	0	0
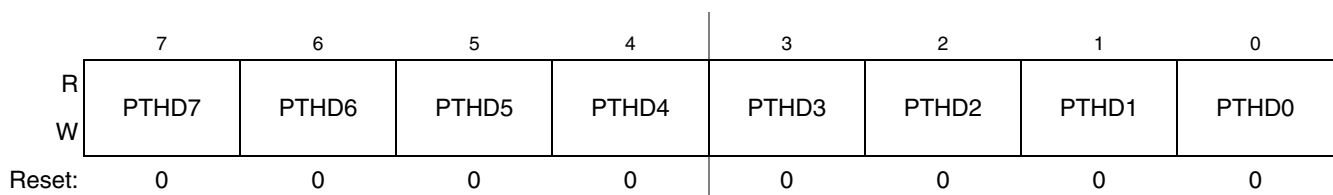
Figure 6-45. Port H Data Register (PTHD)

Table 6-42. PTHD Register Field Descriptions

Field	Description
7–0 PTHD*n*	Port H Data Register Bits. For port H pins configured as inputs, reads return the logic level on the pin. For port H pins configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port H pins configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTHD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pull-ups disabled.

6.7.8.2   Port H Data Direction Register (PTHDD)

	7	6	5	4	3	2	1	0
R W	PTHDD7	PTHDD6	PTHDD5	PTHDD4	PTHDD3	PTHDD2	PTHDD1	PTHDD0
Reset:	0	0	0	0	0	0	0	0
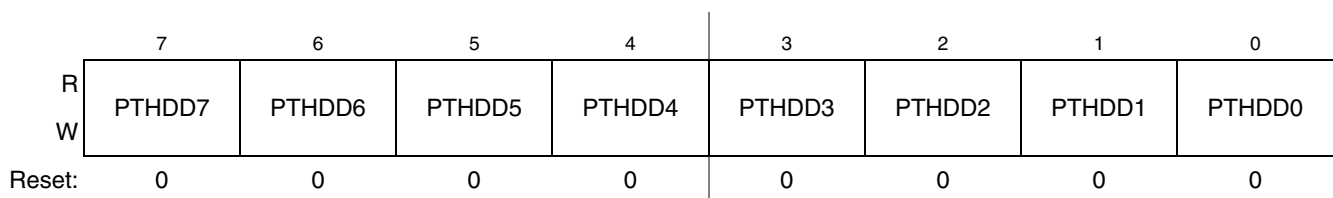
Figure 6-46. Port H Data Direction Register (PTHDD)

Table 6-43. PTHDD Register Field Descriptions

Field	Description
7–0 PTHDD*n*	Data Direction for Port H Bits. These read/write bits control the direction of port H pins and what is read for PTHD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port H bit *n* and PTHD reads return the contents of PTHD*n*.

6.7.8.3　Port H Pull Enable Register (PTHPE)

The port H pull enable register enables pull-ups on the corresponding PTH pin. In some cases, a pull-down device is enabled if pull-downs are supported by an alternate pin function (e.g., KBI).
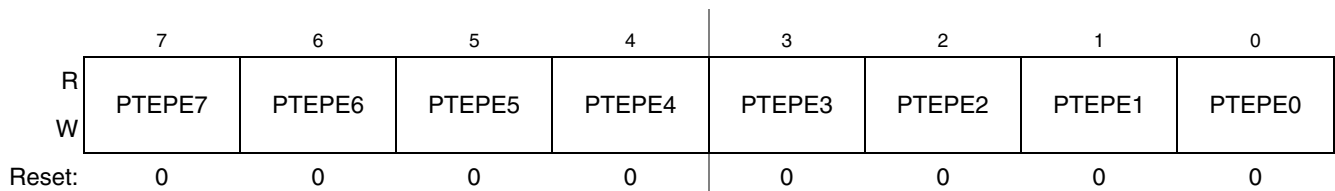
	7	6	5	4	3	2	1	0
R	PTHPE7	PTHPE6	PTHPE5	PTHPE4	PTHPE3	PTHPE2	PTHPE1	PTHPE0
W								
Reset:	0	0	0	0	0	0	0	0
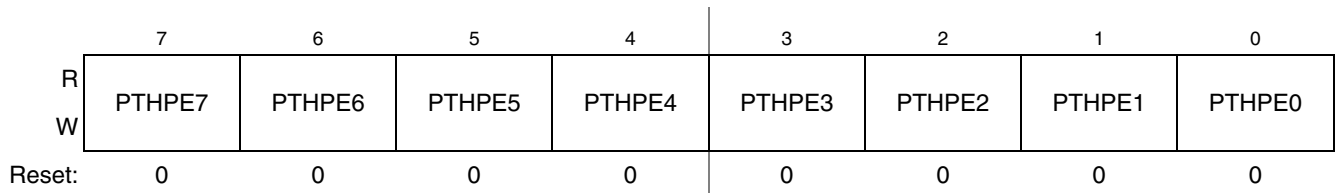
Figure 6-47. Internal Pull Enable for Port H Register (PTHPE)

Table 6-44. PTHPE Register Field Descriptions

Field	Description
7–0 PTHPE*n*	Internal Pull Enable for Port H Bits. Each of these control bits determines if the internal pull-up device is enabled for the associated PTH pin. For port H pins configured as outputs, these bits have no effect and the internal pull devices are disabled. 0　Internal pull-up device disabled for port H bit *n*. 1　Internal pull-up device enabled for port H bit *n*.

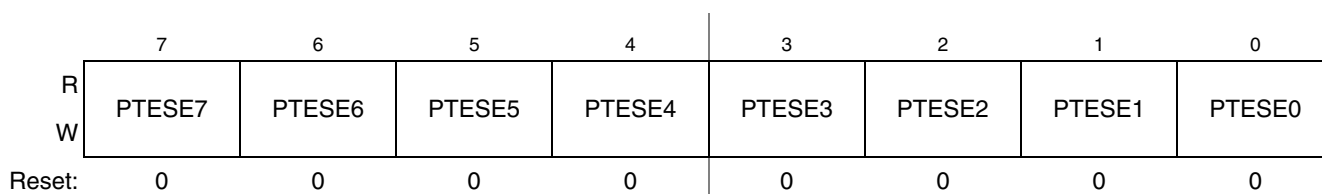## 6.7.8.4　Port H Slew Rate Enable Register (PTHSE)

	7	6	5	4	3	2	1	0
R	PTHSE7	PTHSE6	PTHSE5	PTHSE4	PTHSE3	PTHSE2	PTHSE1	PTHSE0
W								
Reset:	0	0	0	0	0	0	0	0
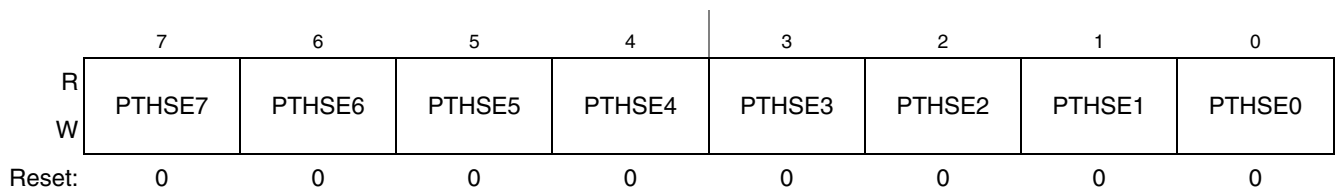
Figure 6-48. Slew Rate Enable for Port H Register (PTHSE)

Table 6-45. PTHSE Register Field Descriptions

Field	Description
7–0 PTHSE*n*	Output Slew Rate Enable for Port H Bits. Each of these control bits determines if the output slew rate control is enabled for the associated PTH pin. For port H pins configured as inputs, these bits have no effect. 0　Output slew rate control disabled for port H bit *n*. 1　Output slew rate control enabled for port H bit *n*.

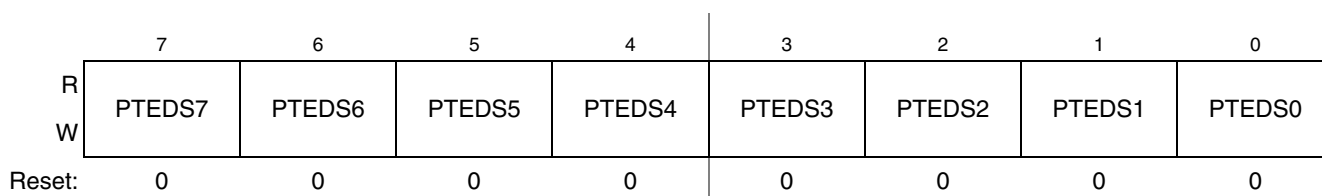## 6.7.8.5　Port H Drive Strength Selection Register (PTHDS)

	7	6	5	4	3	2	1	0
R	PTHDS7	PTHDS6	PTHDS5	PTHDS4	PTHDS3	PTHDS2	PTHDS1	PTHDS0
W								
Reset:	0	0	0	0	0	0	0	0
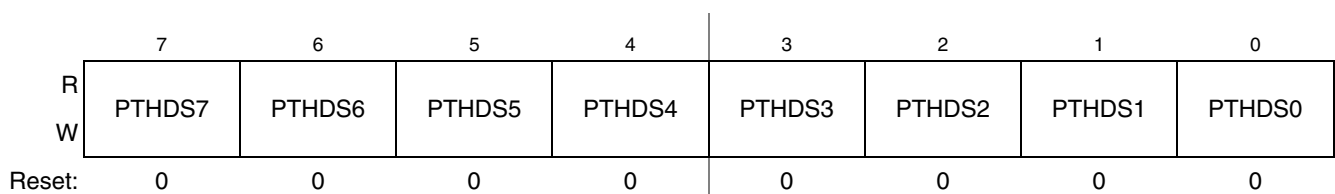
Figure 6-49. Drive Strength Selection for Port H Register (PTHDS)

Table 6-46. PTHDS Register Field Descriptions

Field	Description
7–0 PTHDS*n*	Output Drive Strength Selection for Port H Bits. Each of these control bits selects between low and high output drive for the associated PTH pin. For port H pins configured as inputs, these bits have no effect. 0 Low output drive strength selected for port H bit *n*. 1 High output drive strength selected for port H bit *n*.

6.7.9 Port J Registers

Port J is controlled by the registers listed below.

6.7.9.1 Port J Data Register (PTJD)

	7	6	5	4	3	2	1	0
R W	PTJD7	PTJD6	PTJD5	PTJD4	PTJD3	PTJD2	PTJD1	PTJD0
Reset:	0	0	0	0	0	0	0	0
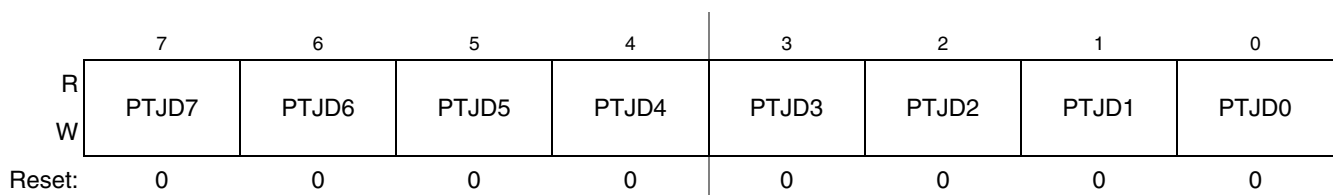
Figure 6-50. Port J Data Register (PTJD)

Table 6-47. PTJD Register Field Descriptions

Field	Description
7–0 PTJD*n*	Port J Data Register Bits. For port J pins configured as inputs, reads return the logic level on the pin. For port J pins configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port J pins configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTJD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pull-ups disabled.

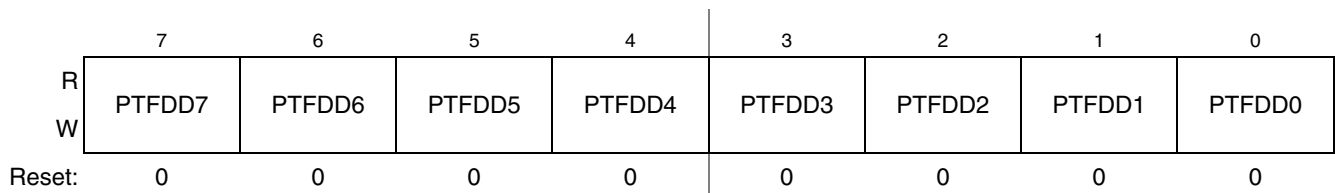### 6.7.9.2 Port J Data Direction Register (PTJDD)

	7	6	5	4	3	2	1	0
R W	PTJDD7	PTJDD6	PTJDD5	PTJDD4	PTJDD3	PTJDD2	PTJDD1	PTJDD0
Reset:	0	0	0	0	0	0	0	0
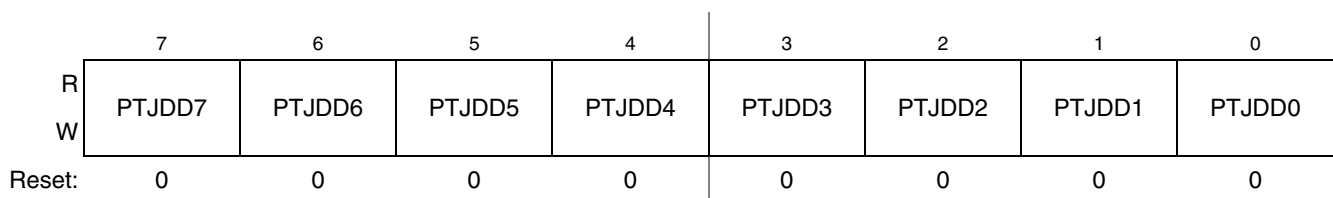
Figure 6-51. Port J Data Direction Register (PTJDD)

Table 6-48. PTJDD Register Field Descriptions

Field	Description
7–0 PTJDD*n*	Data Direction for Port J Bits. These read/write bits control the direction of port J pins and what is read for PTJD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port J bit *n* and PTJD reads return the contents of PTJD*n*.

6.7.9.3 Port J Pull Enable Register (PTJPE)

The port J pull enable register enables pull-ups on the corresponding PTJ pin. In some cases, a pull-down device is enabled if pull-downs are supported by an alternate pin function (e.g., KBI).
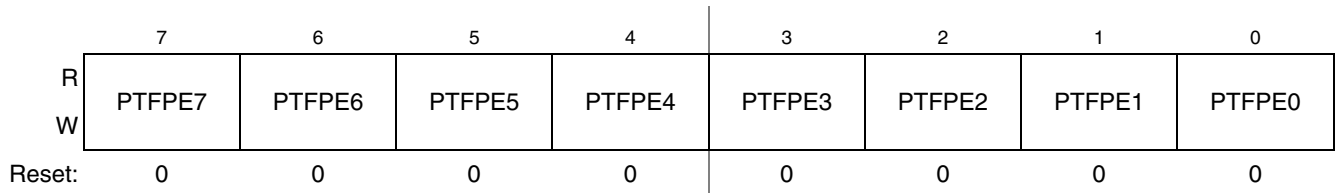
	7	6	5	4	3	2	1	0
R	PTJPE7	PTJPE6	PTJPE5	PTJPE4	PTJPE3	PTJPE2	PTJPE1	PTJPE0
W								
Reset:	0	0	0	0	0	0	0	0
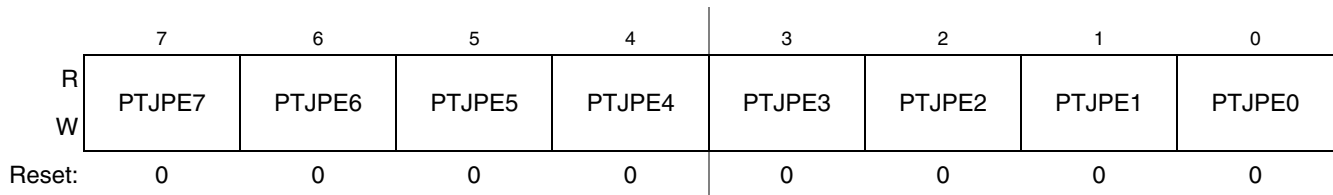
Figure 6-52. Internal Pull Enable for Port J Register (PTJPE)

Table 6-49. PTJPE Register Field Descriptions

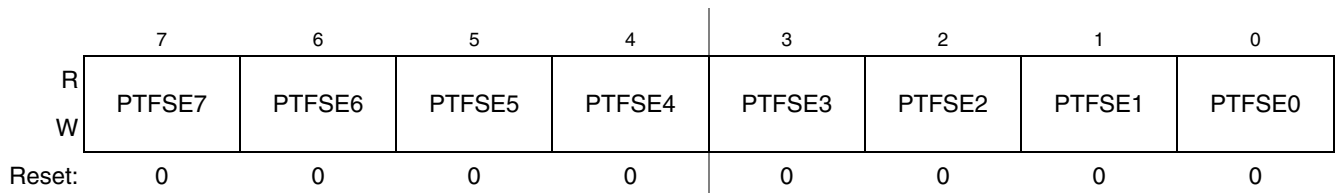| Field	Description
7–0 PTJPE*n*	Internal Pull Enable for Port J Bits. Each of these control bits determines if the internal pull-up device is enabled for the associated PTJ pin. For port J pins configured as outputs, these bits have no effect and the internal pull devices are disabled. 0 Internal pull-up device disabled for port J bit *n*. 1 Internal pull-up device enabled for port J bit *n*.

6.7.9.4 Port J Slew Rate Enable Register (PTJSE)

	7	6	5	4	3	2	1	0
R	PTJSE7	PTJSE6	PTJSE5	PTJSE4	PTJSE3	PTJSE2	PTJSE1	PTJSE0
W								
Reset:	0	0	0	0	0	0	0	0
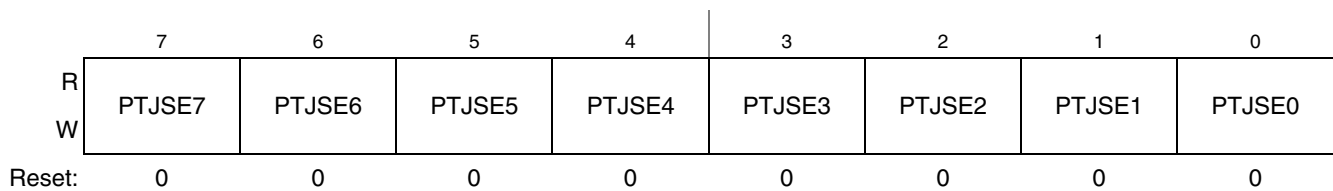
Figure 6-53. Slew Rate Enable for Port J Register (PTJSE)

Table 6-50. PTJSE Register Field Descriptions

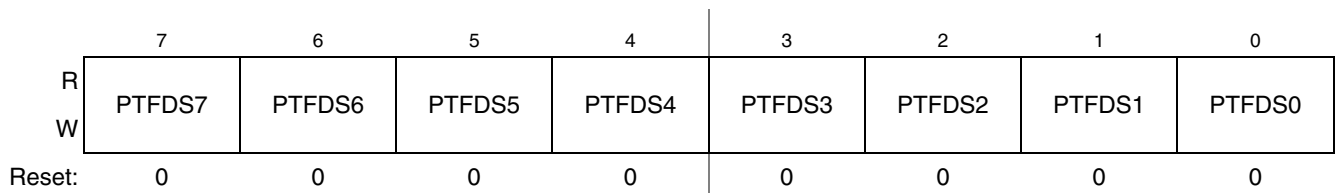| Field	Description
7–0 PTJSE*n*	Output Slew Rate Enable for Port J Bits. Each of these control bits determines if the output slew rate control is enabled for the associated PTJ pin. For port J pins configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port J bit *n*. 1 Output slew rate control enabled for port J bit *n*.

6.7.9.5 Port J Drive Strength Selection Register (PTJDS)

	7	6	5	4	3	2	1	0
R	PTJDS7	PTJDS6	PTJDS5	PTJDS4	PTJDS3	PTJDS2	PTJDS1	PTJDS0
W								
Reset:	0	0	0	0	0	0	0	0
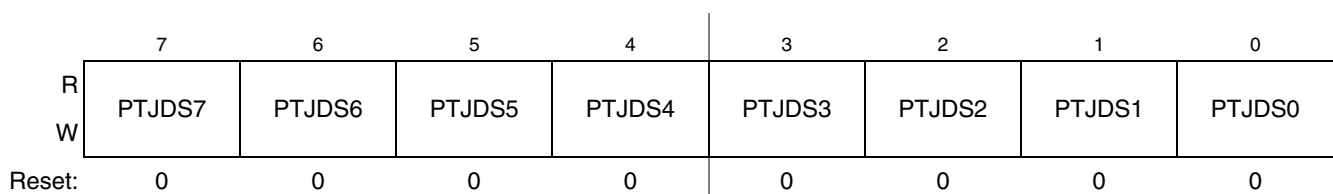
Figure 6-54. Drive Strength Selection for Port J Register (PTJDS)

Table 6-51. PTJDS Register Field Descriptions

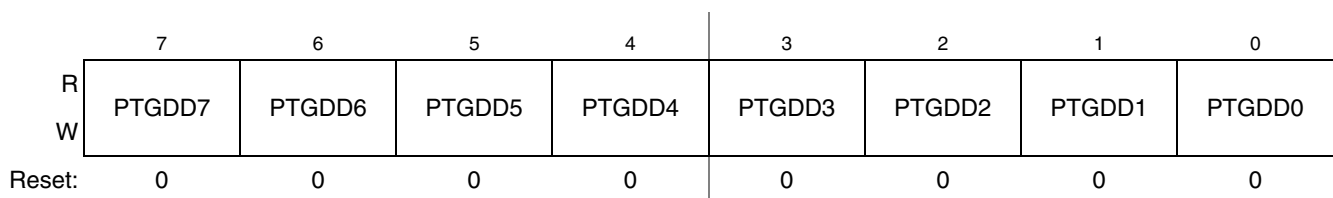| Field	Description
7–0 PTJDS*n*	Output Drive Strength Selection for Port J Bits. Each of these control bits selects between low and high output drive for the associated PTJ pin. For port J pins configured as inputs, these bits have no effect. 0 Low output drive strength selected for port J bit *n*. 1 High output drive strength selected for port J bit *n*.

6.7.10 Keyboard Interrupt 1 (KBI1) Registers

KBI1 is controlled by the registers listed below. Table 6-52 shows KBI1 pin mapping to the port I/O pins.

Table 6-52. KBI1 Pin Mapping

Port pin	PTB3	PTB2	PTB1	PTB0	PTA3	PTA2	PTA1	PTA0
KBI1 pin	KBI1P7	KBI1P6	KBI1P5	KBI1P4	KBI1P3	KBI1P2	KBI1P1	KBI1P0

6.7.10.1 KBI1 Interrupt Status and Control Register (KBI1SC)

	7	6	5	4	3	2	1	0
R	0	0	0	0	KBF	0	KBIE	KBIMOD
W						KBACK		
Reset:	0	0	0	0	0	0	0	0

Figure 6-55. KBI1 Interrupt Status and Control Register (KBI1SC)

Table 6-53. KBI1SC Register Field Descriptions

Field	Description
7–4	Reserved, should be cleared.
3 KBF	KBI1 Interrupt Flag. KBF indicates when a KBI1 interrupt is detected. Writes have no effect on KBF. 0 No KBI1 interrupt detected. 1 KBI1 interrupt detected.
2 KBACK	KBI1 Interrupt Acknowledge. Writing a 1 to KBACK is part of the flag clearing mechanism. KBACK always reads as 0.
1 KBIE	KBI1 Interrupt Enable. KBIE determines whether a KBI1 interrupt is requested. 0 KBI1 interrupt request not enabled. 1 KBI1 interrupt request enabled.
0 KBIMOD	KBI1 Detection Mode. KBIMOD (along with the KBI1ES bits) controls the detection mode of the KBI1 interrupt pins. 0 KBI1 pins detect edges only. 1 KBI1 pins detect edges and levels.

6.7.10.2 KBI1 Interrupt Pin Select Register (KBI1PE)

	7	6	5	4	3	2	1	0
R W	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
Reset:	0	0	0	0	0	0	0	0
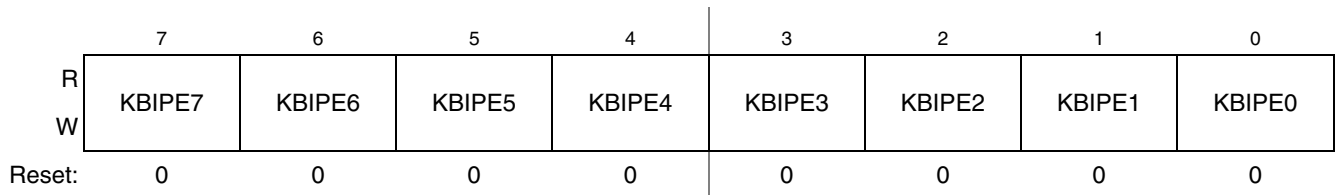
Figure 6-56. KBI1 Interrupt Pin Select Register (KBI1PE)

Table 6-54. KBI1PE Register Field Descriptions

Field	Description
7–0 KBIPE*n*	KBI1 Interrupt Pin Selects. Each of the KBIPE*n* bits enable the corresponding KBI1 interrupt pin. 0 Pin not enabled as interrupt. 1 Pin enabled as interrupt.

6.7.10.3 KBI1 Interrupt Edge Select Register (KBI1ES)

	7	6	5	4	3	2	1	0
R W	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBEDG3	KBEDG2	KBEDG1	KBEDG0
Reset:	0	0	0	0	0	0	0	0
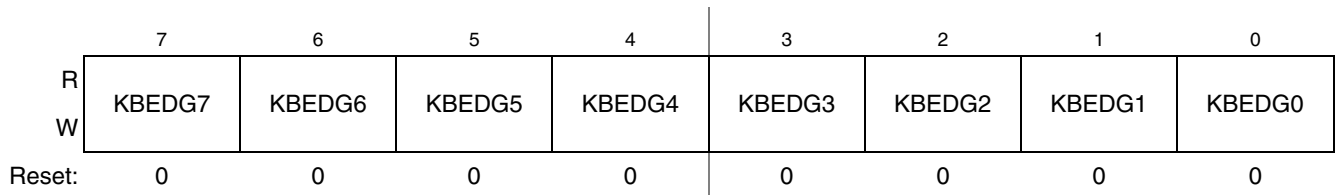
Figure 6-57. KBI1 Edge Select Register (KBI1ES)

Table 6-55. KBI1ES Register Field Descriptions

Field	Description
7–0 KBEDG*n*	KBI1 Edge Selects. Each of the KBEDG*n* bits serves a dual purpose by selecting the polarity of the active interrupt edge as well as selecting a pull-up or pull-down device if enabled. 0 A pull-up device is connected to the associated pin and detects falling edge/low level for interrupt generation. 1 A pull-down device is connected to the associated pin and detects rising edge/high level for interrupt generation.

6.7.11 Keyboard Interrupt 1 (KBI2) Registers

KBI2 is controlled by the registers listed below. Table 6-56 shows KBI2 pin mapping to the port I/O pins.

Table 6-56. KBI2 Pin Mapping

Port pin	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
KBI2 pin	KBI2P7	KBI2P6	KBI2P5	KBI2P4	KBI2P3	KBI2P2	KBI2P1	KBI2P0

6.7.11.1 KBI2 Interrupt Status and Control Register (KBI2SC)

	7	6	5	4	3	2	1	0
R	0	0	0	0	KBF	0	KBIE	KBIMOD
W						KBACK		
Reset:	0	0	0	0	0	0	0	0
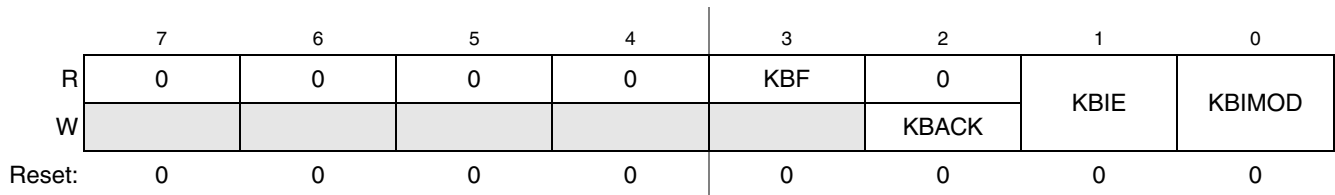
Figure 6-58. KBI2 Interrupt Status and Control Register (KBI2SC)

Table 6-57. KBI2SC Register Field Descriptions

Field	Description
7–4	Reserved, should be cleared.
3 KBF	KBI2 Interrupt Flag. KBF indicates when a KBI2 interrupt is detected. Writes have no effect on KBF. 0 No KBI2 interrupt detected. 1 KBI2 interrupt detected.
2 KBACK	KBI2 Interrupt Acknowledge. Writing a 1 to KBACK is part of the flag clearing mechanism. KBACK always reads as 0.
1 KBIE	KBI2 Interrupt Enable. KBIE determines whether a KBI2 interrupt is requested. 0 KBI2 interrupt request not enabled. 1 KBI2 interrupt request enabled.
0 KBIMOD	KBI2 Detection Mode. KBIMOD (along with the KBI2ES bits) controls the detection mode of the KBI2 interrupt pins. 0 KBI2 pins detect edges only. 1 KBI2 pins detect edges and levels.

6.7.11.2 KBI2 Interrupt Pin Select Register (KBI2PE)

	7	6	5	4	3	2	1	0
R	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
W								
Reset:	0	0	0	0	0	0	0	0
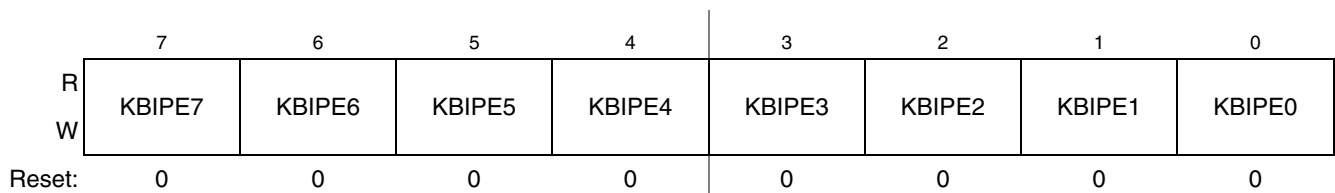
Figure 6-59. KBI2 Interrupt Pin Select Register (KBI2PE)

Table 6-58. KBI2PE Register Field Descriptions

Field	Description
7–0 KBIPE*n*	KBI2 Interrupt Pin Selects. Each of the KBIPE*n* bits enable the corresponding KBI2 interrupt pin. 0 Pin not enabled as interrupt. 1 Pin enabled as interrupt.

6.7.11.3 KBI2 Interrupt Edge Select Register (KBI2ES)

	7	6	5	4	3	2	1	0
R W	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBEDG3	KBEDG2	KBEDG1	KBEDG0
Reset:	0	0	0	0	0	0	0	0
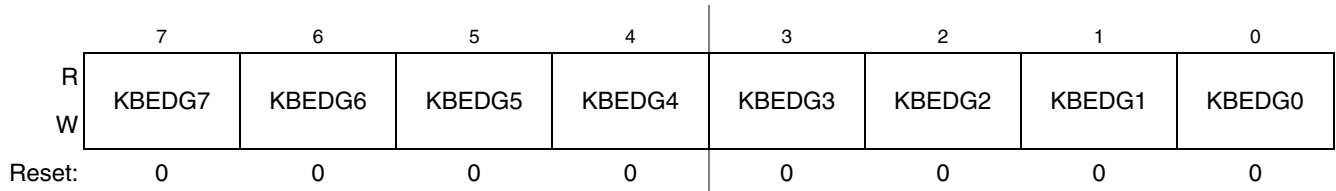
Figure 6-60. KBI2 Edge Select Register (KBI2ES)

Table 6-59. KBI2ES Register Field Descriptions

Field	Description
7–0 KBEDG*n*	KBI2 Edge Selects. Each of the KBEDG*n* bits serves a dual purpose by selecting the polarity of the active interrupt edge as well as selecting a pull-up or pull-down device if enabled. 0 A pull-up device is connected to the associated pin and detects falling edge/low level for interrupt generation. 1 A pull-down device is connected to the associated pin and detects rising edge/high level for interrupt generation.

Chapter 7
ColdFire Core

7.1   Introduction

This section describes the organization of the Version 1 (V1) ColdFire® processor core and an overview of the program-visible registers. For detailed information on instructions, see the ISA_C definition in the *ColdFire Family Programmer's Reference Manual*.

7.1.1   Overview

As with all ColdFire cores, the V1 ColdFire core is comprised of two separate pipelines decoupled by an instruction buffer.
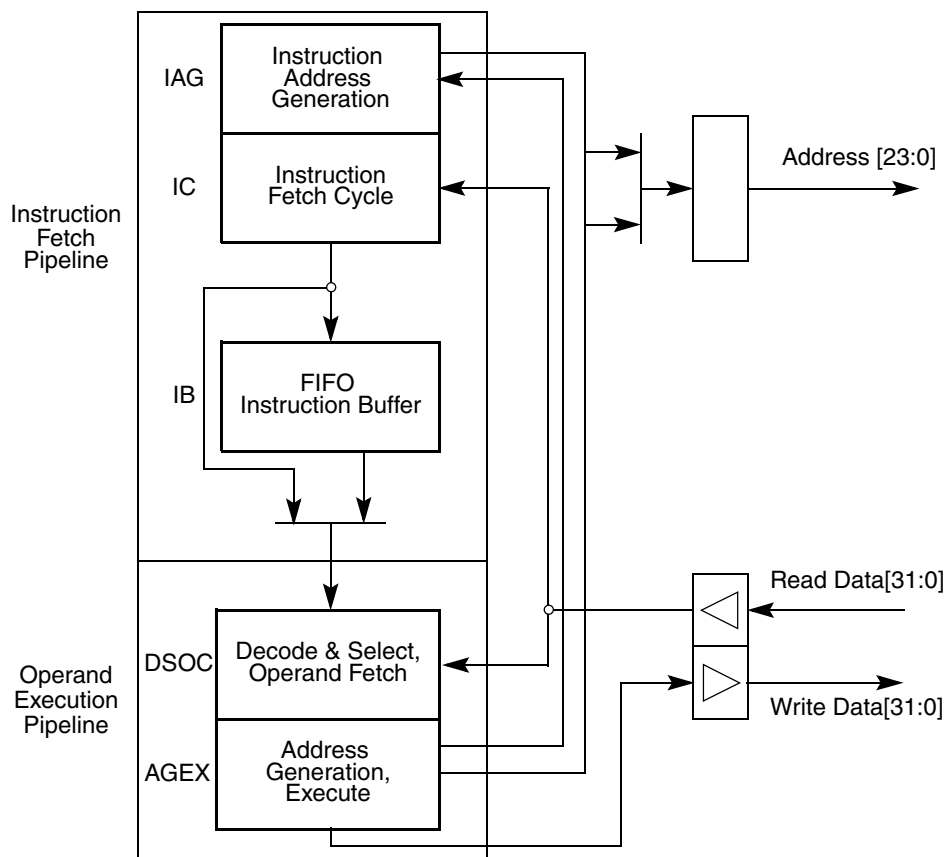
Figure 7-1. V1 ColdFire Core Pipelines

The instruction fetch pipeline (IFP) is a two-stage pipeline for prefetching instructions. The prefetched instruction stream is then gated into the two-stage operand execution pipeline (OEP), which decodes the

instruction, fetches the required operands and then executes the required function. Because the IFP and OEP pipelines are decoupled by an instruction buffer serving as a FIFO queue, the IFP is able to prefetch instructions in advance of their actual use by the OEP thereby minimizing time stalled waiting for instructions.

The V1 ColdFire core pipeline stages include the following:

- Two-stage instruction fetch pipeline (IFP) (plus optional instruction buffer stage)
  - — Instruction address generation (IAG) — Calculates the next prefetch address
  - — Instruction fetch cycle (IC)—Initiates prefetch on the processor's local bus
  - — Instruction buffer (IB) — Optional buffer stage minimizes fetch latency effects using FIFO queue
- Two-stage operand execution pipeline (OEP)
  - — Decode and select/operand fetch cycle (DSOC)—Decodes instructions and fetches the required components for effective address calculation, or the operand fetch cycle
  - — Address generation/execute cycle (AGEX)—Calculates operand address or executes the instruction

When the instruction buffer is empty, opcodes are loaded directly from the IC cycle into the operand execution pipeline. If the buffer is not empty, the IFP stores the contents of the fetched instruction in the IB until it is required by the OEP. The instruction buffer on the V1 core contains three longwords of storage.

For register-to-register and register-to-memory store operations, the instruction passes through both OEP stages once. For memory-to-register and read-modify-write memory operations, an instruction is effectively staged through the OEP twice: the first time to calculate the effective address and initiate the operand fetch on the processor's local bus, and the second time to complete the operand reference and perform the required function defined by the instruction.

The resulting pipeline and local bus structure allow the V1 ColdFire core to deliver sustained high performance across a variety of demanding embedded applications.

7.2   Memory Map/Register Description

The following sections describe the processor registers in the user and supervisor programming models. The programming model is selected based on the processor privilege level (user mode or supervisor mode) as defined by the S bit of the status register (SR). Table 7-1 lists the processor registers.

The user-programming model consists of the following registers:

- 16 general-purpose 32-bit registers (D0–D7, A0–A7)
- 32-bit program counter (PC)
- 8-bit condition code register (CCR)

The supervisor-programming model is intended to be used only by system control software to implement restricted operating system functions, I/O control, and memory management. All accesses that affect the control features of ColdFire processors are in the supervisor programming model, which consists of registers available in user mode as well as the following control registers:

- 16-bit status register (SR)
- 32-bit supervisor stack pointer (SSP)
- 32-bit vector base register (VBR)
- 32-bit CPU configuration register (CPUCR)

Table 7-1. ColdFire Core Programming Model

BDM Command[1]	Register	Width (bits)	Access	Reset Value	Written with MOVEC[2]	Section/Page
Supervisor/User Access Registers						
Load: 0x60 Store: 0x40	Data Register 0 (D0)	32	R/W	0xCF10_0029	No	7.2.1/7-147
Load: 0x61 Store: 0x41	Data Register 1 (D1)	32	R/W	0x1090_0050	No	7.2.1/7-147
Load: 0x62–7 Store: 0x42–7	Data Register 2–7 (D2–D7)	32	R/W	POR: Undefined Else: Unaffected	No	7.2.1/7-147
Load: 0x68–8E Store: 0x48–8E	Address Register 0–6 (A0–A6)	32	R/W	POR: Undefined Else: Unaffected	No	7.2.2/7-148
Load: 0x6F Store: 0x4F	User A7 Stack Pointer (A7)	32	R/W	POR: Undefined Else: Unaffected	No	7.2.3/7-148
Load: 0xEE Store: 0xCE	Condition Code Register (CCR)	8	R/W	POR: Undefined Else: Unaffected	No	7.2.4/7-149
Load: 0xEF Store: 0xCF	Program Counter (PC)	32	R/W	Contents of location 0x0000_0004	No	7.2.5/7-150
Supervisor Access Only Registers						
Load: 0xE0 Store: 0xC0	Supervisor A7 Stack Pointer (OTHER_A7)	32	R/W	Contents of location 0x0000_0000	No	7.2.3/7-148
Load: 0xE1 Store: 0xC1	Vector Base Register (VBR)	32	R/W	0x0000_0000	Yes; Rc = 0x801	7.2.6/7-150
Load: 0xE2 Store: 0xC2	CPU Configuration Register (CPUCR)	32	W	0x0000_0000	Yes; Rc = 0x802	7.2.7/7-151
Load: 0xEE Store: 0xCE	Status Register (SR)	16	R/W	0x27--	No	7.2.8/7-152

[1] The values listed in this column represent the 8-bit BDM command code used when accessing the core registers via the 1-pin BDM port. For more information see ." (These BDM commands are not similar to other ColdFire processors.)

[2] If the given register is written using the MOVEC instruction, the 12-bit control register address (Rc) is also specified.

7.2.1   Data Registers (D0–D7)

D0–D7 data registers are for bit (1-bit), byte (8-bit), word (16-bit) and longword (32-bit) operations; they can also be used as index registers.

NOTE

Registers D0 and D1 contain hardware configuration details after reset. See
Section 7.3.3.14, "Reset Exception" for more details.

BDM: Load: 0x60 + *n; n* = 0-7 (D*n*)
Store: 0x40 + *n; n* = 0-7 (D*n*)

Access: User read/write
BDM read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																	Data															
W																																
Reset (D2-D7)	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset (D0, D1)									See Section 7.3.3.14, "Reset Exception"																							
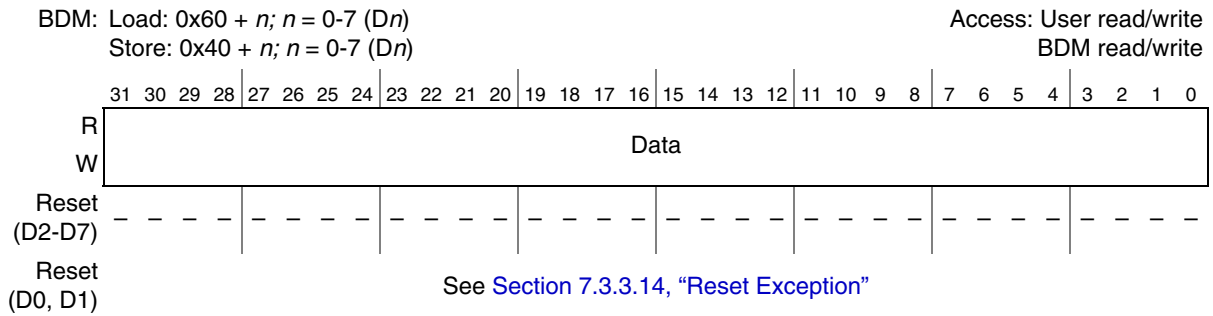
Figure 7-2. Data Registers (D0–D7)

7.2.2 Address Registers (A0–A6)

These registers can be used as software stack pointers, index registers, or base address registers. They can also be used for word and longword operations.

BDM: Load: 0x68 + *n; n* = 0-6 (A*n*)
Store: 0x48 + *n; n* = 0-6 (A*n*)

Access: User read/write
BDM read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																	Address															
W																																
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
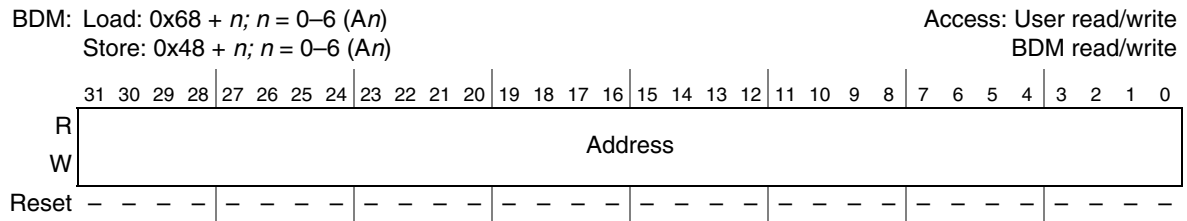
Figure 7-3. Address Registers (A0–A6)

7.2.3 Supervisor/User Stack Pointers (A7 and OTHER_A7)

This ColdFire architecture supports two independent stack pointer (A7) registers—the supervisor stack pointer (SSP) and the user stack pointer (USP). The hardware implementation of these two program-visible 32-bit registers does not identify one as the SSP and the other as the USP. Instead, the hardware uses one 32-bit register as the active A7 and the other as OTHER_A7. Thus, the register contents are a function of the processor operation mode, as shown in the following:

```
if SR[S] = 1

        then    A7 = Supervisor Stack Pointer
                OTHER_A7 = User Stack Pointer
        else    A7 = User Stack Pointer
                OTHER_A7 = Supervisor Stack Pointer
```

The BDM programming model supports direct reads and writes to A7 and OTHER_A7. It is the responsibility of the external development system to determine, based on the setting of SR[S], the mapping of A7 and OTHER_A7 to the two program-visible definitions (SSP and USP).

To support dual stack pointers, the following two supervisor instructions are included in the ColdFire instruction set architecture to load/store the USP:

```
move.l Ay,USP;move to USP
move.l USP,Ax;move from USP
```

These instructions are described in the *ColdFire Family Programmer's Reference Manual*. All other instruction references to the stack pointer, explicit or implicit, access the active A7 register.

NOTE

The USP must be initialized using the `move.l Ay,USP` instruction before any entry into user mode.

The SSP is loaded during reset exception processing with the contents of location 0x(00)00_0000.

BDM: Load: 0x6F (A7)   Access: A7: User or BDM read/write
Store: 0x4F (A7)   OTHER_A7: Supervisor or BDM read/write
Load: 0xE0 (OTHER_A7)
Store: 0xC0 (OTHER_A7)

	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
R					Address			
W								
Reset	– – – –	– – – –	– – – –	– – – –	– – – –	– – – –	– – – –	– – – –
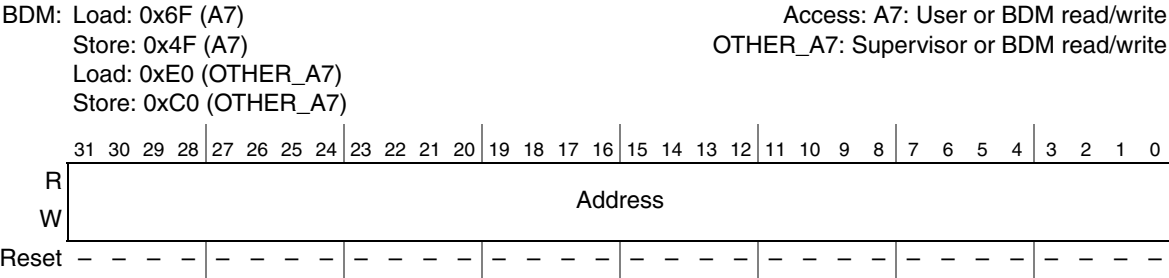
Figure 7-4. Stack Pointer Registers (A7 and OTHER_A7)

7.2.4 Condition Code Register (CCR)

The CCR is the LSB of the processor status register (SR). Bits 4–0 act as indicator flags for results generated by processor operations. The extend bit (X) is also an input operand during multiprecision arithmetic computations. The CCR register must be explicitly loaded after reset and before any compare (CMP), Bcc, or Scc instructions are executed.

BDM: LSB of Status Register (SR)   Access: User read/write
Load: 0xEE (SR)   BDM read/write
Store: 0xCE (SR)

	7	6	5	4	3	2	1	0
R	0	0	0	X	N	Z	V	C
W								
Reset:	0	0	0	—	—	—	—	—
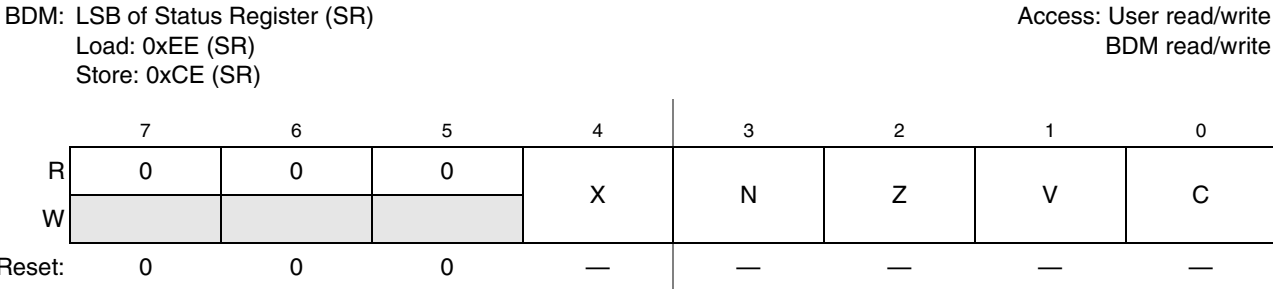
Figure 7-5. Condition Code Register (CCR)

Table 7-2. CCR Field Descriptions

Field	Description
7–5	Reserved, must be cleared.
4 X	Extend condition code bit. Set to the C-bit value for arithmetic operations; otherwise not affected or set to a specified result.
3 N	Negative condition code bit. Set if most significant bit of the result is set; otherwise cleared.
2 Z	Zero condition code bit. Set if result equals zero; otherwise cleared.
1 V	Overflow condition code bit. Set if an arithmetic overflow occurs implying the result cannot be represented in operand size; otherwise cleared.
0 C	Carry condition code bit. Set if a carry out of the operand msb occurs for an addition or if a borrow occurs in a subtraction; otherwise cleared.

7.2.5   Program Counter (PC)

The PC contains the currently executing instruction address. During instruction execution and exception processing, the processor automatically increments contents of the PC or places a new value in the PC, as appropriate. The PC is a base address for PC-relative operand addressing.

The PC is initially loaded during reset exception processing with the contents of location 0x(00)00_0004.

BDM: Load: 0xEF (PC)   Access: User read/write
Store: 0xCF (PC)   BDM read/write

	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
R W				Address				
Reset	– – – –	– – – –	– – – –	– – – –	– – – –	– – – –	– – – –	– – – –
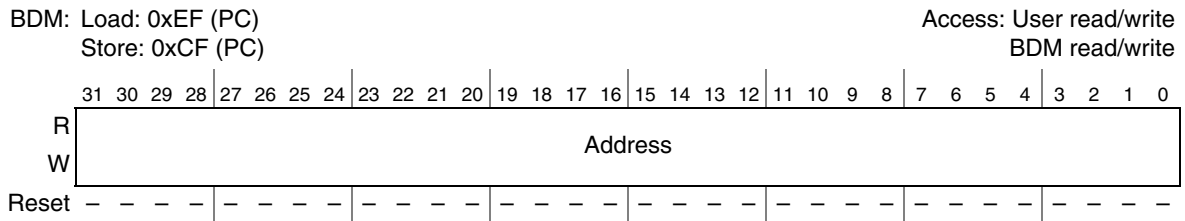
Figure 7-6. Program Counter Register (PC)

7.2.6   Vector Base Register (VBR)

The VBR contains the base address of the exception vector table in memory. To access the vector table, the displacement of an exception vector is added to the value in VBR. The lower 20 bits of the VBR are not implemented by ColdFire processors. They are assumed to be zero, forcing the table to be aligned on a 1 MByte boundary.

In addition, because the V1 ColdFire core supports a 16 Mbyte address space, the upper byte of the VBR is also forced to zero. The VBR can be used to relocate the exception vector table from its default position in the flash memory (address 0x(00)00_0000) to the base of the RAM (address 0x(00)80_0000) if needed.

BDM: 0x801 (VBR)
Load: 0xE1 (VBR)
Store: 0xC1 (VBR)

Access: Supervisor read/write
BDM read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	Base Address				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
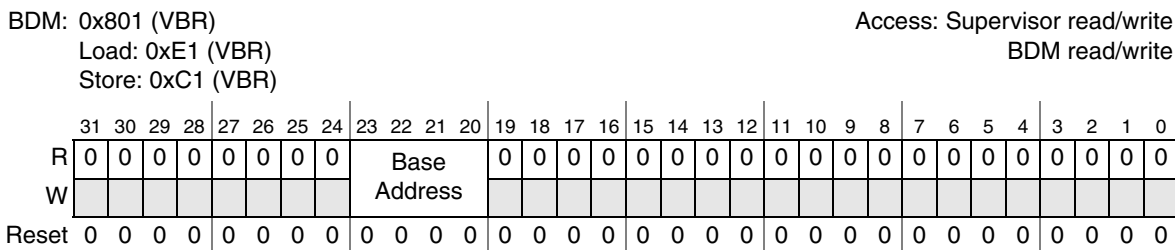
Figure 7-7. Vector Base Register (VBR)

7.2.7 CPU Configuration Register (CPUCR)

The CPUCR provides supervisor mode configurability of specific core functionality. Certain hardware features can be enabled/disabled individually based on the state of the CPUCR.

BDM: 0x802 (CPUCR)
Load: 0xE2 (CPUCR)
Store: 0xC2 (CPUCR)

Access: Supervisor read/write
BDM read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ARD	IRD	IAE	IME	BWD	0	FSD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
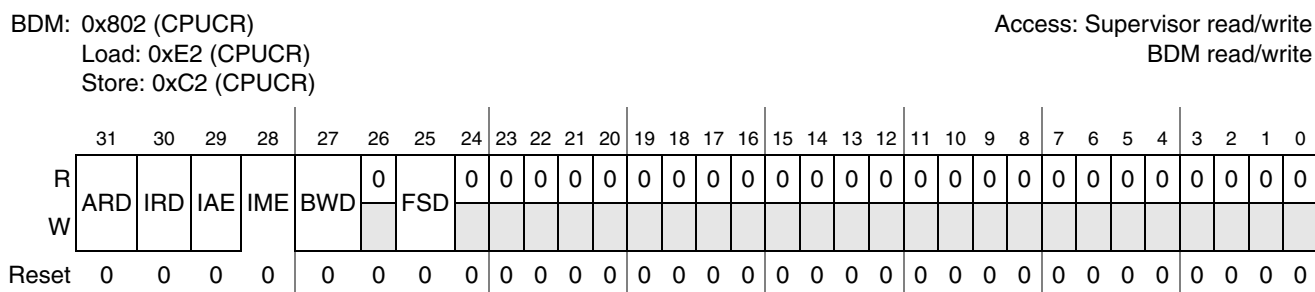
Figure 8. CPU Configuration Register (CPUCR)

Table 3. CPUCR Field Descriptions

Field	Description
31 ARD	Address-related reset disable. Used to disable the generation of a reset event in response to a processor exception caused by an address error, a bus error, an RTE format error, or a fault-on-fault halt condition. 0 The detection of these types of exception conditions or the fault-on-fault halt condition generate a reset event. 1 No reset is generated in response to these exception conditions.
30 IRD	Instruction-related reset disable. Used to disable the generation of a reset event in response to a processor exception caused by the attempted execution of an illegal instruction (except for the ILLEGAL opcode), illegal line A, illegal line F instructions, or a privilege violation. 0 The detection of these types of exception conditions generate a reset event. 1 No reset is generated in response to these exception conditions.
29 IAE	Interrupt acknowledge (IACK) enable. Forces the processor to generate an IACK read cycle from the interrupt controller during exception processing to retrieve the vector number of the interrupt request being acknowledged. The processor's execution time for an interrupt exception is slightly improved when this bit is cleared. 0 The processor uses the vector number provided by the interrupt controller at the time the request is signaled. 1 IACK read cycle from the interrupt controller is generated.
28 IME	Interrupt mask enable. Forces the processor to raise the interrupt level mask (SR[I]) to 7 during every interrupt exception. This capability is provided to assist when porting S08 application code to ColdFire. 0 As part of an interrupt exception, the processor sets SR[I] to the level of the interrupt being serviced. 1 As part of an interrupt exception, the processor sets SR[I] to 7. This disables all level 1-6 interrupt requests but allows recognition of the edge-sensitive level 7 requests.

MCF51QE128 MCU Series Reference Manual, Rev. 3

Table 3. CPUCR Field Descriptions (continued)

Field	Description
27 BWD	Buffered peripheral bus write disable. 0 Peripheral bus writes are buffered and the bus cycle is terminated immediately and does not wait for the peripheral bus termination status. 1 Disables the buffering of peripheral bus writes and does not terminate the bus cycle until a registered version of the peripheral bus termination is received. **Note:** If buffered writes are enabled (BWD = 0), any peripheral bus error status is lost.
26	Reserved, should be cleared.
25 FSD	Flash speculation disabled. Disables certain performance-enhancing features related to address speculation in the flash memory controller. 0 The flash controller tries to speculate on read accesses to improve processor performance by minimizing the exposed flash memory access time. Recall the basic flash access time is two processor cycles. 1 Certain flash address speculation is disabled.
24–0	Reserved, should be cleared.

7.2.8 Status Register (SR)

The SR stores the processor status and includes the CCR, the interrupt priority mask, and other control bits. In supervisor mode, software can access the entire SR. In user mode, only the lower 8 bits (CCR) are accessible. The control bits indicate the following states for the processor: trace mode (T bit), supervisor or user mode (S bit), and master or interrupt state (M bit). All defined bits in the SR have read/write access when in supervisor mode. The lower byte of the SR (the CCR) must be loaded explicitly after reset and before any compare (CMP), Bcc, or Scc instructions execute.

BDM: Load: 0xEE (SR)   Access: Supervisor read/write
      Store: 0xCE (SR)   BDM read/write

		System Byte						Condition Code Register (CCR)							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R T	0	S	M	0		I		0	0	0	X	N	Z	V	C
W															
Reset 0	0	1	0	0	1	1	1	0	0	0	—	—	—	—	—
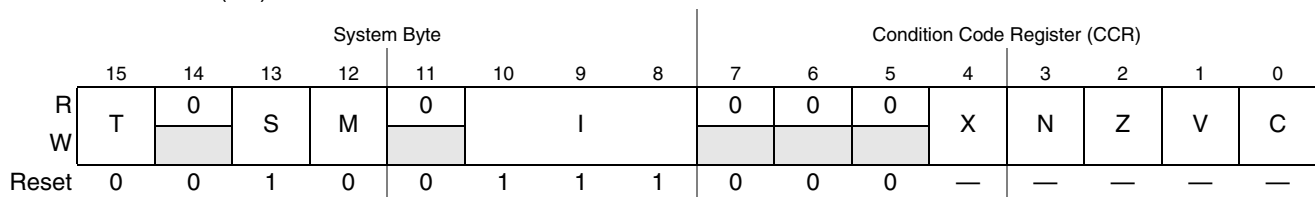
Figure 7-9. Status Register (SR)

Table 7-4. SR Field Descriptions

Field	Description
15 T	Trace enable. When set, the processor performs a trace exception after every instruction.
14	Reserved, must be cleared.
13 S	Supervisor/user state. 0 User mode 1 Supervisor mode
12 M	Master/interrupt state. Bit is cleared by an interrupt exception and software can set it during execution of the RTE or move to SR instructions.
11	Reserved, must be cleared.

Table 7-4. SR Field Descriptions (continued)

Field	Description
10–8 I	Interrupt level mask. Defines current interrupt level. Interrupt requests are inhibited for all priority levels less than or equal to current level, except edge-sensitive level 7 requests, which cannot be masked.
7–0 CCR	Refer to Section 7.2.4, "Condition Code Register (CCR)".

7.3 Functional Description

7.3.1 Instruction Set Architecture (ISA_C)

The original ColdFire Instruction Set Architecture (ISA_A) was derived from the M68000 family opcodes based on extensive analysis of embedded application code. The ISA was optimized for code compiled from high-level languages where the dominant operand size was the 32-bit integer declaration. This approach minimized processor complexity and cost, while providing excellent performance for compiled applications.

After the initial ColdFire compilers were created, developers noted there were certain ISA additions that would enhance code density and overall performance. Additionally, as users implemented ColdFire-based designs into a wide range of embedded systems, they found certain frequently-used instruction sequences that could be improved by the creation of additional instructions.

The original ISA definition minimized support for instructions referencing byte- and word-sized operands. Full support for the move byte and move word instructions was provided, but the only other opcodes supporting these data types are clr (clear) and tst (test). A set of instruction enhancements has been implemented in subsequent ISA revisions, ISA_B and ISA_C. The new opcodes primarily addressed three areas:

1. Enhanced support for byte and word-sized operands
2. Enhanced support for position-independent code
3. Miscellaneous instruction additions to address new functionality

Table 7-5 summarizes the instructions added to revision ISA_A to form revision ISA_C. For more details see the *ColdFire Family Programmer's Reference Manual*.

Table 7-5. Instruction Enhancements over Revision ISA_A

Instruction	Description
BITREV	The contents of the destination data register are bit-reversed; that is, new Dn[31] equals old Dn[0], new Dn[30] equals old Dn[1], ..., new Dn[0] equals old Dn[31].
BYTEREV	The contents of the destination data register are byte-reversed; that is, new Dn[31:24] equals old Dn[7:0], ..., new Dn[7:0] equals old Dn[31:24].
FF1	The data register, Dn, is scanned, beginning from the most-significant bit (Dn[31]) and ending with the least-significant bit (Dn[0]), searching for the first set bit. The data register is then loaded with the offset count from bit 31 where the first set bit appears.
MOV3Q.L	Moves 3-bit immediate data to the destination location.

Table 7-5. Instruction Enhancements over Revision ISA_A (continued)

Instruction	Description
Move from USP	User Stack Pointer → Destination register
Move to USP	Source register → User Stack Pointer
MVS.{B,W}	Sign-extends source operand and moves it to destination register.
MVZ.{B,W}	Zero-fills source operand and moves it to destination register.
SATS.L	Performs saturation operation for signed arithmetic and updates destination register, depending on CCR[V] and bit 31 of the register.
TAS.B	Performs indivisible read-modify-write cycle to test and set addressed memory byte.
Bcc.L	Branch conditionally, longword
BSR.L	Branch to sub-routine, longword
CMP.{B,W}	Compare, byte and word
CMPA.W	Compare address, word
CMPI.{B,W}	Compare immediate, byte and word
MOVEI	Move immediate, byte and word to memory using Ax with displacement
STLDSR	Pushes the contents of the status register onto the stack and then reloads the status register with the immediate data value.

7.3.2   Exception Processing Overview

Exception processing for ColdFire processors is streamlined for performance. The ColdFire processors differ from the M68000 family because they include:

- A simplified exception vector table
- Reduced relocation capabilities using the vector-base register
- A single exception stack frame format
- Use of separate system stack pointers for user and supervisor modes.

All ColdFire processors use an instruction restart exception model. Exception processing includes all actions from fault condition detection to the initiation of fetch for first handler instruction. Exception processing is comprised of four major steps:

1. The processor makes an internal copy of the SR and then enters supervisor mode by setting the S bit and disabling trace mode by clearing the T bit. The interrupt exception also forces the M bit to be cleared and the interrupt priority mask to set to current interrupt request level.

2. The processor determines the exception vector number. For all faults except interrupts, the processor performs this calculation based on exception type. For interrupts, the processor performs an interrupt-acknowledge (IACK) bus cycle to obtain the vector number from the interrupt controller if CPUCR[IAE] is set. The IACK cycle is mapped to special locations within the interrupt controller's address space with the interrupt level encoded in the address. If CPUCR[IAE] is cleared, the processor uses the vector number supplied by the interrupt controller at the time the request was signaled for improved performance.

3. The processor saves the current context by creating an exception stack frame on the system stack. The exception stack frame is created at a 0-modulo-4 address on top of the system stack pointed to by the supervisor stack pointer (SSP). As shown in Figure 7-10, the processor uses a simplified fixed-length stack frame for all exceptions. The exception type determines whether the program counter placed in the exception stack frame defines the location of the faulting instruction (fault) or the address of the next instruction to be executed (next).

4. The processor calculates the address of the first instruction of the exception handler. By definition, the exception vector table is aligned on a 1 Mbyte boundary. This instruction address is generated by fetching an exception vector from the table located at the address defined in the vector base register. The index into the exception table is calculated as ($4 \times$ vector number). After the exception vector has been fetched, the vector contents determine the address of the first instruction of the desired handler. After the instruction fetch for the first opcode of the handler has initiated, exception processing terminates and normal instruction processing continues in the handler.

All ColdFire processors support a 1024-byte vector table aligned on any 1 Mbyte address boundary (see Table 7-6). For the V1 ColdFire core, the only practical locations for the vector table are based at 0x(00)00_0000 in the flash or 0x(00)80_0000 in the RAM. The table contains 256 exception vectors; the first 64 are defined for the core and the remaining 192 are device-specific peripheral interrupt vectors. See Chapter 8, "Interrupt Controller (CF1_INTC)" for details on the device-specific interrupt sources.

For the V1 ColdFire core, the table is partially populated with the first 64 reserved for internal processor exceptions, while vectors 64-102 are reserved for the peripheral I/O requests and the seven software interrupts. Vectors 103–255 are unused and reserved.

Table 7-6. Exception Vector Assignments

Vector Number(s)	Vector Offset (Hex)	Stacked Program Counter	Assignment
0	0x000	—	Initial supervisor stack pointer
1	0x004	—	Initial program counter
2	0x008	Fault	Access error
3	0x00C	Fault	Address error
4	0x010	Fault	Illegal instruction
5–7	0x014–0x01C	—	Reserved
8	0x020	Fault	Privilege violation
9	0x024	Next	Trace
10	0x028	Fault	Unimplemented line-A opcode
11	0x02C	Fault	Unimplemented line-F opcode
12	0x030	Next	Debug interrupt
13	0x034	—	Reserved
14	0x038	Fault	Format error
15–23	0x03C–0x05C	—	Reserved
24	0x060	Next	Spurious interrupt

Table 7-6. Exception Vector Assignments (continued)

Vector Number(s)	Vector Offset (Hex)	Stacked Program Counter	Assignment
25–31	0x064–0x07C	—	Reserved
32–47	0x080–0x0BC	Next	Trap # 0-15 instructions
48–60	0x0C0–0x0F0	—	Reserved
61	0x0F4	Fault	Unsupported instruction
62–63	0x0F8–0x0FC	—	Reserved
64–95	0x100–0x17C	Next	Device-specific interrupts
96–102	0x180–0x198	Next	Level 1–7 software interrupts
103–255	0x19C–0x3FC	—	Reserved, unused for V1

[1] Fault refers to the PC of the instruction that caused the exception. Next refers to the PC of the instruction that follows the instruction that caused the fault.

All ColdFire processors inhibit interrupt sampling during the first instruction of all exception handlers. This allows any handler to disable interrupts effectively, if necessary, by raising the interrupt mask level contained in the status register. In addition, the ISA_C architecture includes an instruction (STLDSR) that stores the current interrupt mask level and loads a value into the SR. This instruction is specifically intended for use as the first instruction of an interrupt service routine that services multiple interrupt requests with different interrupt levels. Finally, the V1 ColdFire core includes the CPUCR[IME] bit that forces the processor to automatically raise the mask level to 7 during the interrupt exception, removing the need for any explicit instruction in the service routine to perform this function. For more details, see *ColdFire Family Programmer's Reference Manual*.

7.3.2.1 Exception Stack Frame Definition

Figure 7-10 shows exception stack frame. The first longword contains the 16-bit format/vector word (F/V) and the 16-bit status register, and the second longword contains the 32-bit program counter address.

	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
SSP →	Format	FS[3:2]	Vector		FS[1:0]	Status Register		
+ 0x4	Program Counter							
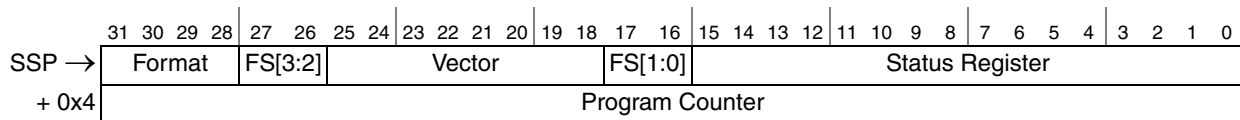
Figure 7-10. Exception Stack Frame Form

The 16-bit format/vector word contains three unique fields:

- A 4-bit format field at the top of the system stack is always written with a value of 4, 5, 6, or 7 by the processor, indicating a two-longword frame format. See Table 7-7.

Table 7-7. Format Field Encodings

Original SSP @ Time of Exception, Bits 1:0	SSP @ 1st Instruction of Handler	Format Field
00	Original SSP - 8	0100
01	Original SSP - 9	0101
10	Original SSP - 10	0110
11	Original SSP - 11	0111

- There is a 4-bit fault status field, FS[3:0], at the top of the system stack. This field is defined for access and address errors only and written as zeros for all other exceptions. See Table 7-8.

Table 7-8. Fault Status Encodings

FS[3:0]	Definition
00xx	Reserved
0100	Error on instruction fetch
0101	Reserved
011x	Reserved
1000	Error on operand write
1001	Attempted write to write-protected space
101x	Reserved
1100	Error on operand read
1101	Reserved
111x	Reserved

- The 8-bit vector number, vector[7:0], defines the exception type and is calculated by the processor for all internal faults and represents the value supplied by the interrupt controller in case of an interrupt. See Table 7-6.

7.3.2.2   S08 and ColdFire Exception Processing Comparison

This section presents a brief summary comparing the exception processing differences between the S08 and V1 ColdFire processor families.

Table 7-9. Exception Processing Comparison

Attribute	S08	V1 ColdFire
Exception Vector Table	32, 2-byte entries, fixed location at upper end of memory	103, 4-byte entries, located at lower end of memory at reset, relocatable with the VBR
More on Vectors	2 for CPU + 30 for IRQs, reset at upper address	64 for CPU + 39 for IRQs, reset at lowest address
Exception Stack Frame	5-byte frame: CCR, A, X, PC	8-byte frame: F/V, SR, PC; General-purpose registers (An, Dn) must be saved/restored by the ISR
Interrupt Levels	1 = f(CCR[I])	7 = f(SR[I]) with automatic hardware support for nesting
Non-Maskable IRQ Support	No	Yes, with level 7 interrupts
Core-enforced IRQ Sensitivity	No	Level 7 is edge sensitive, else level sensitive
INTC Vectoring	Fixed priorities and vector assignments	Fixed priorities and vector assignments, plus any 2 IRQs can be remapped as the highest priority level 6 requests
Software IACK	No	Yes
Exit Instruction from ISR	RTI	RTE

The notion of a software IACK refers to the ability to query the interrupt controller near the end of an interrupt service routine (after the current interrupt request has been cleared) to determine if there are any pending (but currently masked) interrupt requests. If the response to the software IACK's byte operand read is non-zero, the service routine uses the value as the vector number of the highest pending interrupt request and passes control to the appropriate new handler. This process avoids the overhead of a context restore and RTE instruction execution followed immediately by another interrupt exception and context save. In system environments with high rates of interrupt activity, this mechanism can improve overall performance noticeably.

Emulation of the S08's 1-level IRQ processing can easily be managed by software convention within the ColdFire interrupt service routines. For this type of operation, only two of the seven interrupt levels are used:

- SR[I] equals 0 indicates interrupts are enabled
- SR[I] equals 7 indicates interrupts are disabled

Recall that ColdFire treats true level 7 interrupts as edge-sensitive, non-maskable requests. Typically, only the IRQ input pin and a low-voltage detect are assigned as level 7 requests. All the remaining interrupt requests (levels 1-6) are masked when SR[I] equals 7. In any case, all ColdFire processors guarantee that the first instruction of any exception handler is executed before interrupt sampling resumes. By making the first instruction of the ISR a store/load status register (`STLDSR #0x2700`) or a move-to-SR (`MOVE.W #2700,SR`) instruction, interrupts can be safely disabled until the service routine is exited with an RTE instruction that lowers the SR[I] back to level 0. The same functionality can also be provided without an explicit instruction by setting CPUCR[IME] since this forces the processor to load SR[I] with 7 on each interrupt exception.

7.3.3 Processor Exceptions

7.3.3.1 Access Error Exception

The default operation of the V1 ColdFire processor is the generation of an illegal address reset event if an access error (also known as a bus error) is detected. If CPUCR[ARD] is set, the reset is disabled and a processor exception is generated as detailed below.

The exact processor response to an access error depends on the memory reference being performed. For an instruction fetch, the processor postpones the error reporting until the faulted reference is needed by an instruction for execution. Therefore, faults during instruction prefetches followed by a change of instruction flow do not generate an exception. When the processor attempts to execute an instruction with a faulted opword and/or extension words, the access error is signaled and the instruction aborted. For this type of exception, the programming model has not been altered by the instruction generating the access error.

If the access error occurs on an operand read, the processor immediately aborts the current instruction's execution and initiates exception processing. In this situation, any address register updates attributable to the auto-addressing modes, (for example, (An)+,-(An)), have already been performed, so the programming model contains the updated An value. In addition, if an access error occurs during a MOVEM instruction loading from memory, any registers already updated before the fault occurs contain the operands from memory.

The V1 ColdFire processor uses an imprecise reporting mechanism for access errors on operand writes. Because the actual write cycle may be decoupled from the processor's issuing of the operation, the signaling of an access error appears to be decoupled from the instruction that generated the write. Accordingly, the PC contained in the exception stack frame merely represents the location in the program when the access error was signaled. All programming model updates associated with the write instruction are completed. The NOP instruction can collect access errors for writes. This instruction delays its execution until all previous operations, including all pending write operations, are complete. If any previous write terminates with an access error, it is guaranteed to be reported on the NOP instruction.

7.3.3.2 Address Error Exception

The default operation of the V1 ColdFire processor is the generation of an illegal address reset event if an address error is detected. If CPUCR[ARD] equals 1, then the reset is disabled and a processor exception is generated as detailed below.

Any attempted execution transferring control to an odd instruction address (that is, if bit 0 of the target address is set) results in an address error exception.

Any attempted use of a word-sized index register (Xn.w) or a scale factor of eight on an indexed effective addressing mode generates an address error, as does an attempted execution of a full-format indexed addressing mode, which is defined by bit 8 of extension word 1 being set.

If an address error occurs on an RTS instruction, the Version 1 ColdFire processor overwrites the faulting return PC with the address error stack frame.

7.3.3.3   Illegal Instruction Exception

The default operation of the V1 ColdFire processor is the generation of an illegal opcode reset event if an illegal instruction is detected. If CPUCR[IRD] is set, the reset is disabled and a processor exception is generated as detailed below. There is one special case involving the ILLEGAL opcode (0x4AFC); attempted execution of this instruction always generates an illegal instruction exception, regardless of the state of the CPUCR[IRD] bit.

The ColdFire variable-length instruction set architecture supports three instruction sizes: 16, 32, or 48 bits. The first instruction word is known as the operation word (or opword), while the optional words are known as extension word 1 and extension word 2. The opword is further subdivided into three sections: the upper four bits segment the entire ISA into 16 instruction lines, the next 6 bits define the operation mode (opmode), and the low-order 6 bits define the effective address. See Figure 7-11. The opword line definition is shown in Table 7-10.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Line				OpMode						Effective Address					
										Mode		Register			
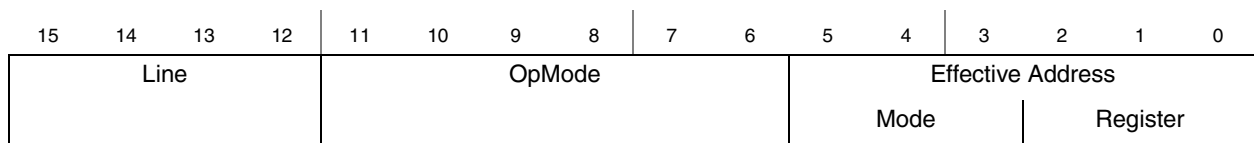
Figure 7-11. ColdFire Instruction Operation Word (Opword) Format

Table 7-10. ColdFire Opword Line Definition

Opword[Line]	Instruction Class
0x0	Bit manipulation, Arithmetic and Logical Immediate
0x1	Move Byte
0x2	Move Long
0x3	Move Word
0x4	Miscellaneous
0x5	Add (ADDQ) and Subtract Quick (SUBQ), Set according to Condition Codes (Scc)
0x6	PC-relative change-of-flow instructions Conditional (Bcc) and unconditional (BRA) branches, subroutine calls (BSR)
0x7	Move Quick (MOVEQ), Move with sign extension (MVS) and zero fill (MVZ)
0x8	Logical OR (OR)
0x9	Subtract (SUB), Subtract Extended (SUBX)
0xA	Move 3-bit Quick (MOV3Q)
0xB	Compare (CMP), Exclusive-OR (EOR)
0xC	Logical AND (AND), Multiply Word (MUL)
0xD	Add (ADD), Add Extended (ADDX)
0xE	Arithmetic and logical shifts (ASL, ASR, LSL, LSR)
0xF	Write DDATA (WDDATA), Write Debug (WDEBUG)

In the original M68000 ISA definition, lines A and F were effectively reserved for user-defined operations (line A) and co-processor instructions (line F). Accordingly, there are two unique exception vectors associated with illegal opwords in these two lines.

Any attempted execution of an illegal 16-bit opcode (except for line-A and line-F opcodes) generates an illegal instruction exception (vector 4). Additionally, any attempted execution of any line-A and most line-F opcodes generate their unique exception types, vector numbers 10 and 11, respectively. ColdFire cores do not provide illegal instruction detection on the extension words on any instruction, including MOVEC.

The V1 ColdFire processor also detects two special cases involving illegal instruction conditions:

1. If execution of the stop instruction is attempted and neither low-power stop nor wait modes are enabled, the processor signals an illegal instruction.

2. If execution of the halt instruction is attempted and BDM is not enabled (XCSR[ENBDM] equals 0), the processor signals an illegal instruction.

In both cases, the processor response is then dependent on the state of CPUCR[IRD]— a reset event or a processor exception.

7.3.3.4 Privilege Violation

The default operation of the V1 ColdFire processor is the generation of an illegal opcode reset event if a privilege violation is detected. If CPUCR[IRD] is set, the reset is disabled and a processor exception is generated as detailed below.

The attempted execution of a supervisor mode instruction while in user mode generates a privilege violation exception. See *ColdFire Programmer's Reference Manual* for a list of supervisor-mode instructions.

There is one special case involving the HALT instruction. Normally, this opcode is a supervisor mode instruction, but if the debug module's CSR[UHE] is set, then this instruction can be also be executed in user mode for debugging purposes.

7.3.3.5 Trace Exception

To aid in program development, all ColdFire processors provide an instruction-by-instruction tracing capability. While in trace mode, indicated by setting of the SR[T] bit, the completion of an instruction execution (for all but the stop instruction) signals a trace exception. This functionality allows a debugger to monitor program execution.

The stop instruction has the following effects:

1. The instruction before the stop executes and then generates a trace exception. In the exception stack frame, the PC points to the stop opcode.

2. When the trace handler is exited, the stop instruction executes, loading the SR with the immediate operand from the instruction.

3. The processor then generates a trace exception. The PC in the exception stack frame points to the instruction after the stop, and the SR reflects the value loaded in the previous step.

If the processor is not in trace mode and executes a stop instruction where the immediate operand sets SR[T], hardware loads the SR and generates a trace exception. The PC in the exception stack frame points to the instruction after the stop, and the SR reflects the value loaded in step 2.

Because ColdFire processors do not support any hardware stacking of multiple exceptions, it is the responsibility of the operating system to check for trace mode after processing other exception types. As an example, consider a TRAP instruction execution while in trace mode. The processor initiates the trap exception and then passes control to the corresponding handler. If the system requires that a trace exception be processed, it is the responsibility of the trap exception handler to check for this condition (SR[T] in the exception stack frame set) and pass control to the trace handler before returning from the original exception.

7.3.3.6   Unimplemented Line-A Opcode

The default operation of the V1 ColdFire processor is the generation of an illegal opcode reset event if an unimplemented line-A opcode is detected. If CPUCR[IRD] is set, the reset is disabled and a processor exception is generated as detailed below.

A line-A opcode is defined when bits 15-12 of the opword are 0b1010. This exception is generated by the attempted execution of an undefined line-A opcode.

7.3.3.7   Unimplemented Line-F Opcode

The default operation of the V1 ColdFire processor is the generation of an illegal opcode reset event if an unimplemented line-F opcode is detected. If CPUCR[IRD] is set, the reset is disabled and a processor exception is generated as detailed below.

A line-F opcode is defined when bits 15-12 of the opword are 0b1111. This exception is generated when attempting to execute an undefined line-F opcode.

7.3.3.8   Debug Interrupt

See Chapter 28, "Debug Module," for a detailed explanation of this exception, which is generated in response to a hardware breakpoint register trigger. The processor does not generate an IACK cycle, but rather calculates the vector number internally (vector number 12). Additionally, SR[M,I] are unaffected by the interrupt.

7.3.3.9   RTE and Format Error Exception

The default operation of the V1 ColdFire processor is the generation of an illegal address reset event if an RTE format error is detected. If CPUCR[ARD] is set, the reset is disabled and a processor exception is generated as detailed below.

When an RTE instruction is executed, the processor first examines the 4-bit format field to validate the frame type. For a ColdFire core, any attempted RTE execution (where the format is not equal to {4,5,6,7}) generates a format error. The exception stack frame for the format error is created without disturbing the original RTE frame and the stacked PC pointing to the RTE instruction.

The selection of the format value provides some limited debug support for porting code from M68000 applications. On M68000 family processors, the SR was located at the top of the stack. On those processors, bit 30 of the longword addressed by the system stack pointer is typically zero. Thus, if an RTE is attempted using this old format, it generates a format error on a ColdFire processor.

If the format field defines a valid type, the processor: (1) reloads the SR operand, (2) fetches the second longword operand, (3) adjusts the stack pointer by adding the format value to the auto-incremented address after the fetch of the first longword, and then (4) transfers control to the instruction address defined by the second longword operand within the stack frame.

7.3.3.10   TRAP Instruction Exception

The TRAP #n instruction always forces an exception as part of its execution and is useful for implementing system calls. The TRAP instruction may be used to change from user to supervisor mode.

This set of 16 instructions provides a similar but expanded functionality compared to the S08's SWI (software interrupt) instruction. These instructions and their functionality should not be confused with the software-scheduled interrupt requests, which are handled like normal I/O interrupt requests by the interrupt controller. The processing of the software-scheduled IRQs can be masked, based on the interrupt priority level defined by the SR[I] field.

7.3.3.11   Unsupported Instruction Exception

If execution of a valid instruction is attempted but the required hardware is not present in the processor, an unsupported instruction exception is generated. The instruction functionality can then be emulated in the exception handler, if desired.

All ColdFire cores record the processor hardware configuration in the D0 register immediately after the negation of $\overline{\text{RESET}}$. See Section 7.3.3.14, "Reset Exception," for details.

For this device, attempted execution of valid integer divide opcodes and all MAC and EMAC instructions result in the unsupported instruction exception.

7.3.3.12   Interrupt Exception

Interrupt exception processing includes interrupt recognition and the fetch of the appropriate vector from the interrupt controller using an IACK cycle or using the previously-supplied vector number, under control of CPUCR[IAE]. See Chapter 8, "Interrupt Controller (CF1_INTC)," for details on the interrupt controller.

7.3.3.13   Fault-on-Fault Halt

The default operation of the V1 ColdFire processor is the generation of an illegal address reset event if a fault-on-fault halt condition is detected. If CPUCR[ARD] is set, the reset is disabled and the processor is halted as detailed below.

If a ColdFire processor encounters any type of fault during the exception processing of another fault, the processor immediately halts execution with the catastrophic fault-on-fault condition. A reset is required to force the processor to exit this halted state.

7.3.3.14   Reset Exception

Asserting the reset input signal ($\overline{RESET}$) to the processor causes a reset exception. The reset exception has the highest priority of any exception; it provides for system initialization and recovery from catastrophic failure. Reset also aborts any processing in progress when the reset input is recognized. Processing cannot be recovered.

The reset exception places the processor in the supervisor mode by setting the SR[S] bit and disables tracing by clearing the SR[T] bit. This exception also clears the SR[M] bit and sets the processor's SR[I] bit to the highest level (level 7, 0b111). Next, the VBR is initialized to zero (0x0000_0000). The control registers specifying the operation of any memories (e.g., cache and/or RAM modules) connected directly to the processor are disabled.

NOTE

> Other implementation-specific registers are also affected. Refer to each module in this reference manual for details on these registers.

After the processor is granted the bus, it performs two longword read-bus cycles. The first longword at address 0x(00)00_0000 is loaded into the supervisor stack pointer and the second longword at address 0x(00)00_0004 is loaded into the program counter. After the initial instruction is fetched from memory, program execution begins at the address in the PC. If an access error or address error occurs before the first instruction is executed, the processor enters the fault-on-fault state.

ColdFire processors load hardware configuration information into the D0 and D1 general-purpose registers after system reset. The hardware configuration information is loaded immediately after the reset-in signal is negated. This allows an emulator to read out the contents of these registers via the BDM to determine the hardware configuration.

Information loaded into D0 defines the processor hardware configuration as shown in .

BDM: Load: 0x60 (D0)   Access: User read-only
        Store: 0x40 (D0)   BDM read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PF								VER				REV			
W																
Reset	1	1	0	0	1	1	1	1	0	0	0	1	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAC	DIV	EMAC	FPU	MMU	0	0	0	ISA				DEBUG			
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1

Figure 7-12. D0 Hardware Configuration Info

Table 7-11. D0 Hardware Configuration Info Field Description

Field	Description
31–24 PF	Processor family. This field is fixed to a hex value of 0xCF indicating a ColdFire core is present.
23–20 VER	ColdFire core version number. Defines the hardware microarchitecture version of ColdFire core. 0001 V1 ColdFire core (This is the value used for this device.) 0010 V2 ColdFire core 0011 V3 ColdFire core 0100 V4 ColdFire core 0101 V5 ColdFire core Else Reserved for future use.
19–16 REV	Processor revision number. The default is 0b0000.
15 MAC	MAC present. This bit signals if the optional multiply-accumulate (MAC) execution engine is present in processor core. 0 MAC execute engine not present in core. (This is the value used for this device.) 1 MAC execute engine is present in core.
14 DIV	Divide present. This bit signals if the hardware divider (DIV) is present in the processor core. 0 Divide execute engine not present in core. (This is the value used for this device.) 1 Divide execute engine is present in core.
13 EMAC	EMAC present. This bit signals if the optional enhanced multiply-accumulate (EMAC) execution engine is present in processor core. 0 EMAC execute engine not present in core. (This is the value used for this device.) 1 EMAC execute engine is present in core.
12 FPU	FPU present. This bit signals if the optional floating-point (FPU) execution engine is present in processor core. 0 FPU execute engine not present in core. (This is the value used for this device.) 1 FPU execute engine is present in core.
11 MMU	MMU present. This bit signals if the optional virtual memory management unit (MMU) is present in processor core. 0 MMU execute engine not present in core. (This is the value used for this device.) 1 MMU execute engine is present in core.
10–8	Reserved.
7–4 ISA	ISA revision. This 4-bit field defines the instruction-set architecture (ISA) revision level implemented in ColdFire processor core. 0000 ISA_A 0001 ISA_B 0010 ISA_C (This is the value used for this device.) 1000 ISA_A+ Else Reserved
3–0 DEBUG	Debug module revision number. This 4-bit field defines revision level of the debug module used in the ColdFire processor core. 0000 DEBUG_A 0001 DEBUG_B 0010 DEBUG_C 0011 DEBUG_D 0100 DEBUG_E 1001 DEBUG_B+ (This is the value used for this device.) 1011 DEBUG_D+ Else Reserved

Information loaded into D1 defines the local memory hardware configuration as shown in the figure below.

BDM: Load: 0x61 (D1)   Access: User read-only
      Store: 0x41 (D1)   BDM read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	1	0	0	0	0		FLASHSZ			0	0	0	0
W																
Reset	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0		SRAMSZ			0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0

Figure 7-13. D1 Hardware Configuration Info

Table 7-12. D1 Hardware Configuration Information Field Description

Field	Description
31–24	Reserved.
23–20 FLASHSZ	Flash bank size. 0000-0111 No flash 1000 64-Kbyte flash 1001 128-Kbyte flash 1010 256-Kbyte flash 1011 512-Kbyte flash Else Reserved for future use.
19–16	Reserved
15–8	Reserved
7–4 SRAMSZ	SRAM bank size. 0000 No SRAM 0001 512 bytes 0010 1 Kbytes 0011 2 Kbytes 0100 4 Kbytes 0101 8 Kbytes (This is the value used for this device) 0110 16 Kbytes 0111 32 Kbytes 1000 64 Kbytes 1001 128 Kbytes Else Reserved for future use
3-0	Reserved.

7.3.4 Instruction Execution Timing

This section presents processor instruction execution times in terms of processor-core clock cycles. The number of operand references for each instruction is enclosed in parentheses following the number of processor clock cycles. Each timing entry is presented as C(R/W) where:

- C is the number of processor clock cycles, including all applicable operand fetches and writes, and all internal core cycles required to complete the instruction execution.
- R/W is the number of operand reads (R) and writes (W) required by the instruction. An operation performing a read-modify-write function is denoted as (1/1).

This section includes the assumptions concerning the timing values and the execution time details.

7.3.4.1 Timing Assumptions

For the timing data presented in this section, these assumptions apply:

1. The OEP is loaded with the opword and all required extension words at the beginning of each instruction execution. This implies that the OEP does not wait for the IFP to supply opwords and/or extension words.

2. The OEP does not experience any sequence-related pipeline stalls. The most common example of stall involves consecutive store operations, excluding the MOVEM instruction. For all STORE operations (except MOVEM), certain hardware resources within the processor are marked as busy for two clock cycles after the final decode and select/operand fetch cycle (DSOC) of the store instruction. If a subsequent STORE instruction is encountered within this 2-cycle window, it is stalled until the resource again becomes available. Thus, the maximum pipeline stall involving consecutive STORE operations is two cycles. The MOVEM instruction uses a different set of resources and this stall does not apply.

3. The OEP completes all memory accesses without any stall conditions caused by the memory itself. Thus, the timing details provided in this section assume that an infinite zero-wait state memory is attached to the processor core.

4. All operand data accesses are aligned on the same byte boundary as the operand size; for example, 16-bit operands aligned on 0-modulo-2 addresses, 32-bit operands aligned on 0-modulo-4 addresses.

   The processor core decomposes misaligned operand references into a series of aligned accesses as shown in Table 7-13.

Table 7-13. Misaligned Operand References

address[1:0]	Size	Bus Operations	Additional C(R/W)
01 or 11	Word	Byte, Byte	2(1/0) if read 1(0/1) if write
01 or 11	Long	Byte, Word, Byte	3(2/0) if read 2(0/2) if write
10	Long	Word, Word	2(1/0) if read 1(0/1) if write

7.3.4.2 MOVE Instruction Execution Times

Table 7-14 lists execution times for MOVE.{B,W} instructions; Table 7-15 lists timings for MOVE.L.

NOTE

For all tables in this section, the execution time of any instruction using the PC-relative effective addressing modes is the same for the comparable An-relative mode.

ET with {<ea> = (d16,PC)}   equals ET with {<ea> = (d16,An)}

ET with {<ea> = (d8,PC,Xi*SF)}   equals ET with {<ea> = (d8,An,Xi*SF)}

The nomenclature xxx.wl refers to both forms of absolute addressing, xxx.w and xxx.l.

Table 7-14. MOVE Byte and Word Execution Times

Source	Destination						
	Rx	(Ax)	(Ax)+	-(Ax)	(d16,Ax)	(d8,Ax,Xi*SF)	xxx.wl
Dy	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)
Ay	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)
(Ay)	2(1/0)	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1))	3(1/1)
(Ay)+	2(1/0)	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1))	3(1/1)
-(Ay)	2(1/0)	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1))	3(1/1)
(d16,Ay)	2(1/0)	3(1/1)	3(1/1)	3(1/1)	3(1/1)	—	—
(d8,Ay,Xi*SF)	3(1/0)	4(1/1)	4(1/1)	4(1/1)	—	—	—
xxx.w	2(1/0)	3(1/1)	3(1/1)	3(1/1)	—	—	—
xxx.l	2(1/0)	3(1/1)	3(1/1)	3(1/1)	—	—	—
(d16,PC)	2(1/0)	3(1/1)	3(1/1)	3(1/1)	3(1/1)	—	—
(d8,PC,Xi*SF)	3(1/0)	4(1/1)	4(1/1)	4(1/1))	—	—	—
#xxx	1(0/0)	3(0/1)	3(0/1)	3(0/1)	1(0/1)	—	—

Table 7-15. MOVE Long Execution Times

Source	Destination						
	Rx	(Ax)	(Ax)+	-(Ax)	(d16,Ax)	(d8,Ax,Xi*SF)	xxx.wl
Dy	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)
Ay	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)
(Ay)	2(1/0)	2(1/1)	2(1/1)	2(1/1)	2(1/1)	3(1/1)	2(1/1)
(Ay)+	2(1/0)	2(1/1)	2(1/1)	2(1/1)	2(1/1)	3(1/1)	2(1/1)
-(Ay)	2(1/0)	2(1/1)	2(1/1)	2(1/1)	2(1/1)	3(1/1)	2(1/1)
(d16,Ay)	2(1/0)	2(1/1)	2(1/1)	2(1/1)	2(1/1)	—	—

Table 7-15. MOVE Long Execution Times (continued)

Source	Destination						
	Rx	**(Ax)**	**(Ax)+**	**-(Ax)**	**(d16,Ax)**	**(d8,Ax,Xi*SF)**	**xxx.wl**
(d8,Ay,Xi*SF)	3(1/0)	3(1/1)	3(1/1)	3(1/1)	—	—	—
xxx.w	2(1/0)	2(1/1)	2(1/1)	2(1/1)	—	—	—
xxx.l	2(1/0)	2(1/1)	2(1/1)	2(1/1)	—	—	—
(d16,PC)	2(1/0)	2(1/1)	2(1/1)	2(1/1)	2(1/1)	—	—
(d8,PC,Xi*SF)	3(1/0)	3(1/1)	3(1/1)	3(1/1)	—	—	—
#xxx	1(0/0)	2(0/1)	2(0/1)	2(0/1)	—	—	—

7.3.4.3 Standard One Operand Instruction Execution Times

Table 7-16. One Operand Instruction Execution Times

Opcode	<EA>	Effective Address							
		Rn	**(An)**	**(An)+**	**-(An)**	**(d16,An)**	**(d8,An,Xn*SF)**	**xxx.wl**	**#xxx**
BITREV	Dx	1(0/0)	—	—	—	—	—	—	—
BYTEREV	Dx	1(0/0)	—	—	—	—	—	—	—
CLR.B	<ea>	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)	—
CLR.W	<ea>	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)	—
CLR.L	<ea>	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)	—
EXT.W	Dx	1(0/0)	—	—	—	—	—	—	—
EXT.L	Dx	1(0/0)	—	—	—	—	—	—	—
EXTB.L	Dx	1(0/0)	—	—	—	—	—	—	—
FF1	Dx	1(0/0)	—	—	—	—	—	—	—
NEG.L	Dx	1(0/0)	—	—	—	—	—	—	—
NEGX.L	Dx	1(0/0)	—	—	—	—	—	—	—
NOT.L	Dx	1(0/0)	—	—	—	—	—	—	—
SATS.L	Dx	1(0/0)	—	—	—	—	—	—	—
SCC	Dx	1(0/0)	—	—	—	—	—	—	—
SWAP	Dx	1(0/0)	—	—	—	—	—	—	—
TAS.B	<ea>	—	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1)	3(1/1)	—
TST.B	<ea>	1(0/0)	2(1/0)	2(1/0)	2(1/0)	2(1/0)	3(1/0)	2(1/0)	1(0/0)
TST.W	<ea>	1(0/0)	2(1/0)	2(1/0)	2(1/0)	2(1/0)	3(1/0)	2(1/0)	1(0/0)
TST.L	<ea>	1(0/0)	2(1/0)	2(1/0)	2(1/0)	2(1/0)	3(1/0)	2(1/0)	1(0/0)

7.3.4.4 Standard Two Operand Instruction Execution Times

Table 7-17. Two Operand Instruction Execution Times

Opcode	<EA>	Effective Address							
		Rn	**(An)**	**(An)+**	**-(An)**	**(d16,An)** **(d16,PC)**	**(d8,An,Xn*SF)** **(d8,PC,Xn*SF)**	**xxx.wl**	**#xxx**
ADD.L	<ea>,Rx	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	1(0/0)
ADD.L	Dy,<ea>	—	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1)	3(1/1)	—
ADDI.L	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
ADDQ.L	#imm,<ea>	1(0/0)	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1)	3(1/1)	—
ADDX.L	Dy,Dx	1(0/0)	—	—	—	—	—	—	—
AND.L	<ea>,Rx	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	1(0/0)
AND.L	Dy,<ea>	—	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1)	3(1/1)	—
ANDI.L	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
ASL.L	<ea>,Dx	1(0/0)	—	—	—	—	—	—	1(0/0)
ASR.L	<ea>,Dx	1(0/0)	—	—	—	—	—	—	1(0/0)
BCHG	Dy,<ea>	2(0/0)	4(1/1)	4(1/1)	4(1/1)	4(1/1)	5(1/1)	4(1/1)	—
BCHG	#imm,<ea>	2(0/0)	4(1/1)	4(1/1)	4(1/1)	4(1/1)	—	—	—
BCLR	Dy,<ea>	2(0/0)	4(1/1)	4(1/1)	4(1/1)	4(1/1)	5(1/1)	4(1/1)	—
BCLR	#imm,<ea>	2(0/0)	4(1/1)	4(1/1)	4(1/1)	4(1/1)	—	—	—
BSET	Dy,<ea>	2(0/0)	4(1/1)	4(1/1)	4(1/1)	4(1/1)	5(1/1)	4(1/1)	—
BSET	#imm,<ea>	2(0/0)	4(1/1)	4(1/1)	4(1/1)	4(1/1)	—	—	—
BTST	Dy,<ea>	2(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	—
BTST	#imm,<ea>	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	—	—	—
CMP.B	<ea>,Rx	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	1(0/0)
CMP.W	<ea>,Rx	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	1(0/0)
CMP.L	<ea>,Rx	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	1(0/0)
CMPI.B	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
CMPI.W	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
CMPI.L	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
EOR.L	Dy,<ea>	1(0/0)	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1)	3(1/1)	—
EORI.L	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
LEA	<ea>,Ax	—	1(0/0)	—	—	1(0/0)	2(0/0)	1(0/0)	—
LSL.L	<ea>,Dx	1(0/0)	—	—	—	—	—	—	1(0/0)
LSR.L	<ea>,Dx	1(0/0)	—	—	—	—	—	—	1(0/0)
MOVEQ.L	#imm,Dx	—	—	—	—	—	—	—	1(0/0)
OR.L	<ea>,Rx	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	1(0/0)
OR.L	Dy,<ea>	—	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1)	3(1/1)	—
ORI.L	#imm,Dx	1(0/0)	—	—	—	—	—	—	—

Table 7-17. Two Operand Instruction Execution Times (continued)

Opcode	<EA>	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An) (d16,PC)	(d8,An,Xn*SF) (d8,PC,Xn*SF)	xxx.wl	#xxx
SUB.L	<ea>,Rx	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	1(0/0)
SUB.L	Dy,<ea>	—	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1)	3(1/1)	—
SUBI.L	#imm,Dx	1(0/0)	—	—	—	—	—	—	—
SUBQ.L	#imm,<ea>	1(0/0)	3(1/1)	3(1/1)	3(1/1)	3(1/1)	4(1/1)	3(1/1)	—
SUBX.L	Dy,Dx	1(0/0)	—	—	—	—	—	—	—
MULS.W	<ea>,Dx	9(0/0)	11(1/0)	11(1/0)	11(1/0)	11(1/0)	12(1/0)	11(1/0)	9(0/0)
MULU.W	<ea>,Dx	9(0/0)	11(1/0)	11(1/0)	11(1/0)	11(1/0)	12(1/0)	11(1/0)	9(0/0)
MULS.L	<ea>,Dx	≤18(0/0)	≤20(1/0)	≤20(1/0)	≤20(1/0)	≤20(1/0)	—	—	—
MULU.L	<ea>,Dx	≤18(0/0)	≤20(1/0)	≤20(1/0)	≤20(1/0)	≤20(1/0)	—	—	—

7.3.4.5 Miscellaneous Instruction Execution Times

Table 7-18. Miscellaneous Instruction Execution Times

Opcode	<EA>	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An)	(d8,An,Xn*SF)	xxx.wl	#xxx
LINK.W	Ay,#imm	2(0/1)	—	—	—	—	—	—	—
MOV3Q.L	#imm,<ea>	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)	—
MOVE.L	Ay,USP	3(0/0)	—	—	—	—	—	—	—
MOVE.L	USP,Ax	3(0/0)	—	—	—	—	—	—	—
MOVE.W	CCR,Dx	1(0/0)	—	—	—	—	—	—	—
MOVE.W	<ea>,CCR	1(0/0)	—	—	—	—	—	—	1(0/0)
MOVE.W	SR,Dx	1(0/0)	—	—	—	—	—	—	—
MOVE.W	<ea>,SR	7(0/0)	—	—	—	—	—	—	7(0/0) [2]
MOVEC	Ry,Rc	9(0/1)	—	—	—	—	—	—	—
MOVEM.L	<ea>,&list	—	1+n(n/0)	—	—	1+n(n/0)	—	—	—
MOVEM.L	&list,<ea>	—	1+n(0/n)	—	—	1+n(0/n)	—	—	—
MVS	<ea>,Dx	1(0/0)	2(1/0)	2(1/0)	2(1/0)	2(1/0)	3(1/0)	2(1/0)	1(0/0)
MVZ	<ea>,Dx	1(0/0)	2(1/0)	2(1/0)	2(1/0)	2(1/0)	3(1/0)	2(1/0)	1(0/0)
NOP		3(0/0)	—	—	—	—	—	—	—
PEA	<ea>	—	2(0/1)	—	—	2(0/1) [4]	3(0/1) [5]	2(0/1)	—
PULSE		1(0/0)	—	—	—	—	—	—	—
STLDSR	#imm	—	—	—	—	—	—	—	5(0/1)
STOP	#imm	—	—	—	—	—	—	—	3(0/0) [3]
TRAP	#imm	—	—	—	—	—	—	—	15(1/2)
TPF		1(0/0)	—	—	—	—	—	—	—

Table 7-18. Miscellaneous Instruction Execution Times (continued)

Opcode	<EA>	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An)	(d8,An,Xn*SF)	xxx.wl	#xxx
TPF.W		1(0/0)	—	—	—	—	—	—	—
TPF.L		1(0/0)	—	—	—	—	—	—	—
UNLK	Ax	2(1/0)	—	—	—	—	—	—	—
WDDATA	<ea>	—	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	—
WDEBUG	<ea>	—	5(2/0)	—	—	5(2/0)	—	—	—

[1]The n is the number of registers moved by the MOVEM opcode.

[2]If a MOVE.W #imm,SR instruction is executed and imm[13] equals 1, the execution time is 1(0/0).

[3]The execution time for STOP is the time required until the processor begins sampling continuously for interrupts.

[4]PEA execution times are the same for (d16,PC).

[5]PEA execution times are the same for (d8,PC,Xn*SF).

7.3.4.6 Branch Instruction Execution Times

Table 7-19. General Branch Instruction Execution Times

Opcode	<EA>	Effective Address							
		Rn	(An)	(An)+	-(An)	(d16,An) (d16,PC)	(d8,An,Xi*SF) (d8,PC,Xi*SF)	xxx.wl	#xxx
BRA		—	—	—	—	2(0/1)	—	—	—
BSR		—	—	—	—	3(0/1)	—	—	—
JMP	<ea>	—	3(0/0)	—	—	3(0/0)	4(0/0)	3(0/0)	—
JSR	<ea>	—	3(0/1)	—	—	3(0/1)	4(0/1)	3(0/1)	—
RTE		—	—	10(2/0)	—	—	—	—	—
RTS		—	—	5(1/0)	—	—	—	—	—

Table 7-20. Bcc Instruction Execution Times

Opcode	Forward Taken	Forward Not Taken	Backward Taken	Backward Not Taken
Bcc	3(0/0)	1(0/0)	2(0/0)	3(0/0)

Chapter 8
Interrupt Controller (CF1_INTC)

8.1   Introduction

This interrupt controller (CF1_INTC) is intended for use in low-cost microcontroller designs using the Version 1 (V1) ColdFire processor core. In keeping with the general philosophy for devices based on this low-end 32-bit processor, the interrupt controller generally supports less programmability compared to similar modules in other ColdFire microcontrollers and embedded microprocessors. However, CF1_INTC provides the required functionality with a minimal silicon cost.

These requirements guide the CF1_INTC module definition to support Freescale's Controller Continuum:

- The priorities of the interrupt requests between comparable HCS08 and V1 ColdFire devices are identical.
- Supports a mode of operation (via software convention with hardware assists) equivalent to the S08's interrupt processing with only one level of nesting.
- Leverages the current ColdFire interrupt controller programming model and functionality, but with a minimal hardware implementation and cost.

Table 8-1 provides a high-level architectural comparison between HCS08 and ColdFire exception processing as these differences are important in the definition of the CF1_INTC module. Throughout this document, the term IRQ refers to an interrupt request and ISR refers to an interrupt service routine to process an interrupt exception.

Table 8-1. Exception Processing Comparison

Attribute	HCS08	V1 ColdFire
Exception Vector Table	32 two-byte entries, fixed location at upper end of memory	103 four-byte entries, located at lower end of memory at reset, relocatable with the VBR
More on Vectors	2 for CPU + 30 for IRQs, reset at upper address	64 for CPU + 39 for IRQs, reset at lowest address
Exception Stack Frame	5-byte frame: CCR, A, X, PC	8-byte frame: F/V, SR, PC; General-purpose registers (An, Dn) must be saved/restored by the ISR
Interrupt Levels	1 = f(CCR[I])	7= f(SR[I]) with automatic hardware support for nesting
Non-Maskable IRQ Support	No	Yes, with level 7 interrupts
Core-enforced IRQ Sensitivity	No	Level 7 is edge sensitive, else level sensitive
INTC Vectoring	Fixed priorities and vector assignments	Fixed priorities and vector assignments, plus any 2 IRQs can be remapped as the highest priority level 6 requests

Table 8-1. Exception Processing Comparison (continued)

Attribute	HCS08	V1 ColdFire
Software IACK	No	Yes
Exit Instruction from ISR	RTI	RTE

8.1.1 Overview

Interrupt exception processing includes interrupt recognition, aborting the current instruction execution stream, storing of an 8-byte exception stack frame in memory, calculation of the appropriate vector, and passing control to the specified interrupt service routine.

Unless specifically noted otherwise, all ColdFire processors sample for interrupts once during each instruction's execution during the first cycle of execution in the OEP. Additionally, all ColdFire processors use an instruction restart exception model.

The ColdFire processor architecture defines a 3-bit interrupt priority mask field in the processor's status register (SR[I]). This field, and the associated hardware, support seven levels of interrupt requests with the processor providing automatic nesting capabilities. The levels are defined in descending numeric order with $7 > 6 \ldots > 1$. Level 7 interrupts are treated as non-maskable, edge-sensitive requests while levels 6–1 are maskable, level-sensitive requests. The SR[I] field defines the processor's current interrupt level. The processor continuously compares the encoded IRQ level from CF1_INTC against SR[I]. Recall that interrupt requests are inhibited for all levels less than or equal to the current level, except the edge-sensitive level 7 request, which cannot be masked.

Exception processing for ColdFire processors is streamlined for performance and includes all actions from the detection of the fault condition to the initiation of fetch for the first handler instruction. Exception processing is comprised of four major steps. The interrupt-specific actions are highlighted.

1. The processor makes an internal copy of the status register (SR) and enters supervisor mode by setting SR[S] and disabling trace mode by clearing SR[T]. Interrupt exception also forces the master mode (M) bit to be cleared and the interrupt priority mask (I) to be set to the level of the current interrupt request.

2. The processor determines the exception vector number. For all faults except interrupts, the processor performs this calculation based on the exception type. For interrupts, the processor performs an IACK bus cycle to obtain the vector number from the interrupt controller if CPUCR[IAE] equals 1. The IACK cycle is mapped to special locations within the interrupt controller's IPS address space with the interrupt level encoded in the address. If CPUCR[IAE] equals 0, the processor uses the vector number supplied by the interrupt controller at the time the request was signaled (for improved performance).

3. The processor saves the current context by creating an exception stack frame on the system stack. As a result, exception stack frame is created at a 0-modulo-4 address on top of the system stack defined by the supervisor stack pointer (SSP). The processor uses an 8-byte stack frame for all exceptions. It contains the vector number of the exception, the contents of the status register at the time of the exception, and the program counter (PC) at the time of the exception. The exception

type determines whether the program counter placed in the exception stack frame defines the location of the faulting instruction (fault) or the address of the next instruction to be executed (next). For interrupts, the stacked PC is always the address of the next instruction to be executed.

- The processor calculates the address of the first instruction of the exception handler. By definition, the exception vector table is aligned on a 1-Mbyte boundary. This instruction address is generated by fetching a 32-bit exception vector from the table located at the address defined in the vector base register (VBR). The index into the exception table is calculated as (4 × vector number). After the exception vector has been fetched, the contents of the vector serves as a 32-bit pointer to the address of the first instruction of the desired handler. After the instruction fetch for the first opcode of the handler has been initiated, exception processing terminates and normal instruction processing continues in the handler.

All ColdFire processors support a 1024-byte vector table aligned on any 1-Mbyte address boundary. For the V1 ColdFire core, the only practical locations for the vector table are based at 0x(00)00_0000 in the flash or 0x(00)80_0000 in the RAM. The table contains 256 exception vectors; the first 64 are reserved for internal processor exceptions, and the remaining 192 are user-defined interrupt vectors. For the V1 ColdFire core, the table is partially populated with the first 64 reserved for internal processor exceptions, while vectors 64 – 102 are reserved for the peripheral I/O requests and the seven software interrupts. The IRQ assignments are device-specific as they depend on the exact set of peripherals for any given device.

A simplified V1 ColdFire exception vector table is shown in Table 8-2.

Table 8-2. V1 ColdFire Exception Vector Table

Vector Number(s)	Vector Offset (Hex)	Stacked Program Counter	Assignment
0	0x000	—	Initial supervisor stack pointer
1	0x004	—	Initial program counter
2–63	0x008–0x0FC	—	Reserved for internal CPU exceptions
64	0x100	Next	IRQ_pin
65	0x104	Next	Low_voltage
66	0x108	Next	TPM1_ch0
67	0x10C	Next	TPM1_ch1
68	0x110	Next	TPM1_ch2
69	0x114	Next	TPM1_ovfl
70	0x118	Next	TPM2_ch0
71	0x11C	Next	TPM2_ch1
72	0x120	Next	TPM2_ch2
73	0x124	Next	TPM2_ovfl
74	0x128	Next	SPI2
75	0x12C	Next	SPI1
76	0x130	Next	SCI1_err

Table 8-2. V1 ColdFire Exception Vector Table (continued)

Vector Number(s)	Vector Offset (Hex)	Stacked Program Counter	Assignment
77	0x134	Next	SCI1_rx
78	0x138	Next	SCI1_tx
79	0x13C	Next	IICx
80	0x140	Next	KBIx
81	0x144	Next	ADC
82	0x148	Next	ACMPx
83	0x14C	Next	SCI2_err
84	0x150	Next	SCI2_rx
85	0x154	Next	SCI2_tx
86	0x158	Next	RTC
87	0x15C	Next	TPM3_ch0
88	0x160	Next	TPM3_ch1
89	0x164	Next	TPM3_ch2
90	0x168	Next	TPM3_ch3
91	0x16C	Next	TPM3_ch4
92	0x170	Next	TPM3_ch5
93	0x174	Next	TPM3_ovfl
94–95	0x178–0x17C	—	Reserved; unused for V1
96	0x180	Next	Level 7 Software Interrupt
97	0x184	Next	Level 6 Software Interrupt
98	0x188	Next	Level 5 Software Interrupt
99	0x18C	Next	Level 4 Software Interrupt
100	0x190	Next	Level 3 Software Interrupt
101	0x194	Next	Level 2 Software Interrupt
102	0x198	Next	Level 1 Software Interrupt
103–255	0x19C–0x3FC	—	Reserved; unused for V1

The basic ColdFire interrupt controller supports up to 63 request sources mapped as nine priorities for each of the seven supported levels (7 levels × 9 priorities per level). Within the nine priorities within a level, the mid-point is typically reserved for package-level IRQ inputs. The levels and priorities within the level follow a descending order: $7 > 6 > ... > 1 > 0$.

The HCS08 architecture supports a 32-entry exception vector table: the first two vectors are reserved for internal CPU/system exceptions and the remaining 30 are available for I/O interrupt requests. The requirement for an exact match between the interrupt requests and priorities across two architectures

means the 30 sources are mapped to a sparsely-populated two-dimensional ColdFire array of seven interrupt levels and nine priorities within the level. The following association between the HCS08 and ColdFire vector numbers applies:

```
ColdFire Vector Number = 62 + HCS08 Vector Number
```

The CF1_INTC performs a cycle-by-cycle evaluation of the active requests and signals the highest-level, highest-priority request to the V1 ColdFire core in the form of an encoded interrupt level and the exception vector associated with the request. The module also includes a byte-wide interface to access its programming model. These interfaces are shown in the simplified block diagram of Figure 8-1.
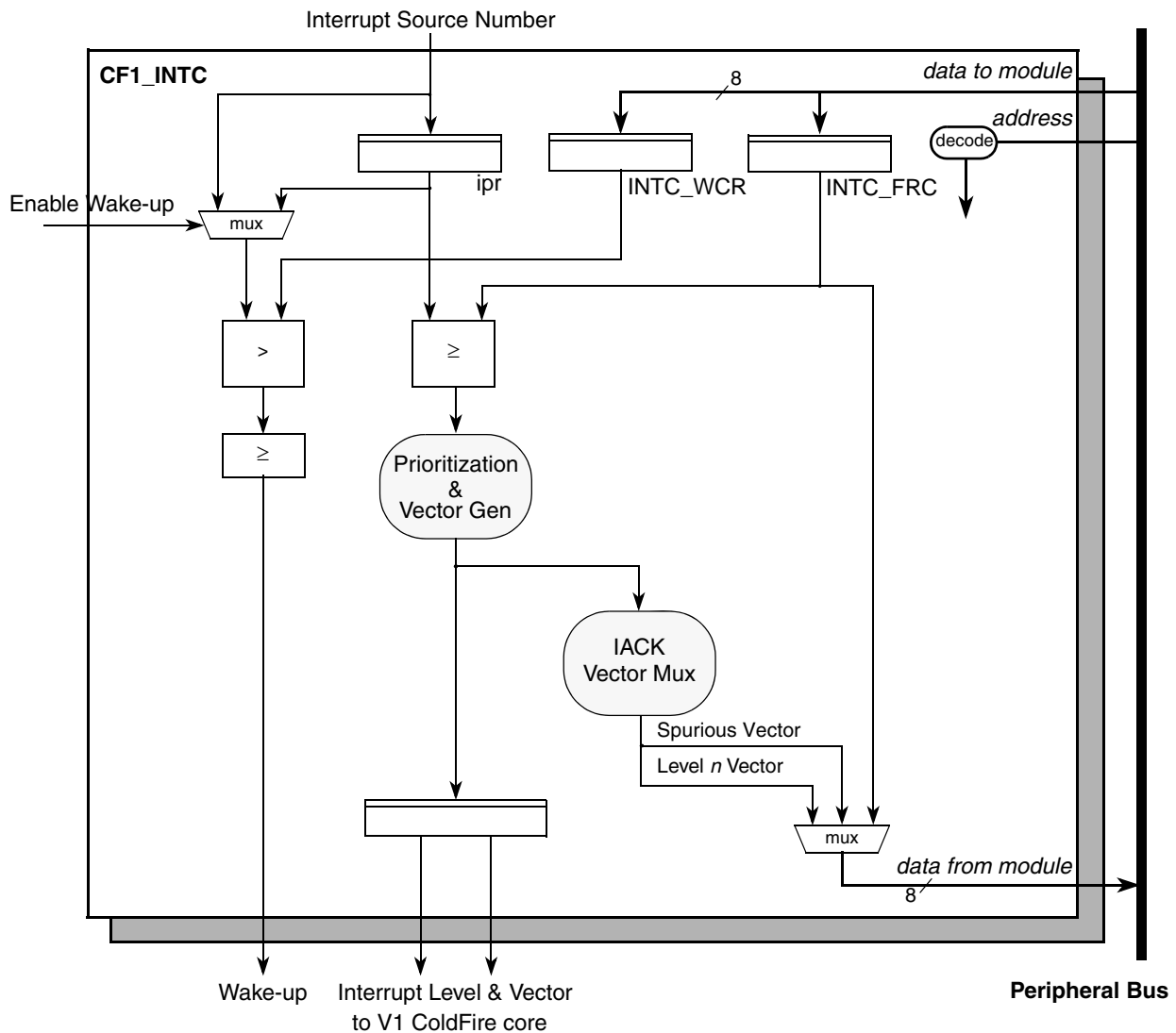
Figure 8-1. CF1_INTC Block Diagram

8.1.2   Features

The Version 1 ColdFire interrupt controller includes:

- Memory-mapped off-platform slave module
  - — 64-byte space located at top end of memory: 0x(FF)FF_FFC0–0x(FF)FF_FFFF
  - — Programming model accessed via the peripheral bus
  - — Encoded interrupt level and vector sent directly to processor core
- Support of 30 peripheral I/O interrupt requests plus seven software (one per level) interrupt requests
- Fixed association between interrupt request source and level plus priority
  - — 30 I/O requests assigned across seven available levels and nine priorities per level
  - — Exactly matches HCS08 interrupt request priorities
  - — Up to two requests can be remapped to the highest maskable level + priority
- Unique vector number for each interrupt source
  - — ColdFire vector number = 62 + HCS08 vector number
  - — Details on IRQ and vector assignments are device-specific
- Support for service routine interrupt acknowledge (software IACK) read cycles for improved system performance
- Combinatorial path provides wake-up signal from wait and sleep modes

8.1.3   Modes of Operation

The CF1_INTC module does not support any special modes of operation. As a memory-mapped slave peripheral located on the platform's slave bus, it responds based strictly on the memory addresses of the connected bus.

One special behavior of the CF1_INTC deserves mention. When the device enters a wait or stop mode of operation and certain clocks are disabled, there is an input signal that can be asserted to enable a purely-combinational logic path for monitoring the assertion of an interrupt request. After a request of unmasked level is asserted, this combinational logic path asserts an output signal which is sent to the clock generation logic to re-enable the internal device clocks to exit the low-power mode.

8.2   External Signal Description

The CF1_INTC module does not include any external interfaces.

8.3   Memory Map and Register Definition

The CF1_INTC module provides a 64-byte programming model mapped to the upper region of the 16 Mbyte address space. All the register names are prefixed with INTC_ as an abbreviation for the full module name.

The programming model is referenced using 8-bit accesses. Attempted references to unimplemented addresses or with a non-supported access type (for example, a write to a read-only register) generate a bus error termination.

The programming model follows the definition from previous ColdFire interrupt controllers. This compatibility accounts for the various memory holes in this module's memory map.

8.3.1 Memory Map

The CF1_INTC module is based at address 0x(FF)FF_FFC0 (referred to as CF1_INTC_BASE throughout the chapter) and occupies the upper 64 bytes of the peripheral space. The module memory map is shown in Table 8-3.

Table 8-3. CF1_INTC Memory Map

Offset Address	Register Name	Register Description	Width (bits)	Access	Reset Value	Section/ Page
0x13	INTC_FRC	CF1_INTC Force Interrupt Register	8	R/W	0x00	8.3.2.1/8-179
0x18	INTC_PL6P7	CF1_INTC Programmable Level 6, Priority 7	8	R/W	0x00	8.3.2.2/8-180
0x19	INTC_PL6P6	CF1_INTC Programmable Level 6, Priority 6	8	R/W	0x00	8.3.2.2/8-180
0x1B	INTC_WCR	CF1_INTC Wake-up Control Register	8	R/W	0x00	8.3.2.3/8-181
0x1E	INTC_SFRC	CF1_INTC Set Interrupt Force Register	8	Write	—	8.3.2.4/8-182
0x1F	INTC_CFRC	CF1_INTC Clear Interrupt Force Register	8	Write	—	8.3.2.5/8-183
0x20	INTC_SWIACK	CF1_INTC Software Interrupt Acknowledge	8	Read	0x00	8.3.2.6/8-184
0x24	INTC_LVL1IACK	CF1_INTC Level 1 Interrupt Acknowledge	8	Read	0x18	8.3.2.6/8-184
0x28	INTC_LVL2IACK	CF1_INTC Level 2 Interrupt Acknowledge	8	Read	0x18	8.3.2.6/8-184
0x2C	INTC_LVL3IACK	CF1_INTC Level 3 Interrupt Acknowledge	8	Read	0x18	8.3.2.6/8-184
0x30	INTC_LVL4IACK	CF1_INTC Level 4 Interrupt Acknowledge	8	Read	0x18	8.3.2.6/8-184
0x34	INTC_LVL15ACK	CF1_INTC Level 5 Interrupt Acknowledge	8	Read	0x18	8.3.2.6/8-184
0x38	INTC_LVL6IACK	CF1_INTC Level 6 Interrupt Acknowledge	8	Read	0x18	8.3.2.6/8-184
0x3C	INTC_LVL7IACK	CF1_INTC Level 7 Interrupt Acknowledge	8	Read	0x18	8.3.2.6/8-184

8.3.2 Register Descriptions

The following sections detail the individual registers within the CF1_INTC's programming model.

8.3.2.1 INTC Force Interrupt Register (INTC_FRC)

The INTC_FRC register allows software to generate a unique interrupt for each possible level at the lowest priority within the level for functional or debug purposes. These interrupts may be self-scheduled by setting one or more of the bits in the INTC_FRC register. In some cases, the handling of a normal interrupt request may cause critical processing by the service routine along with the scheduling (using the INTC_FRC register) of a lower priority level interrupt request to be processed at a later time for less-critical task handling.

The INTC_FRC register may be modified directly using a read-modify-write sequence or through a simple write operation using the set/clear force interrupt registers (INTC_SFRC, INTC_CFRC).

NOTE

Take special notice of the bit numbers within this register, 39–32. This is for compatibility with previous ColdFire interrupt controllers.

Offset: CF1_INTC_BASE + 0x13 (INTC_FRC)   Access: Read/Write

	39	38	37	36	35	34	33	32
R	0	LVL1	LVL2	LVL3	LVL4	LVL5	LVL6	LVL7
W								
Reset	0	0	0	0	0	0	0	0
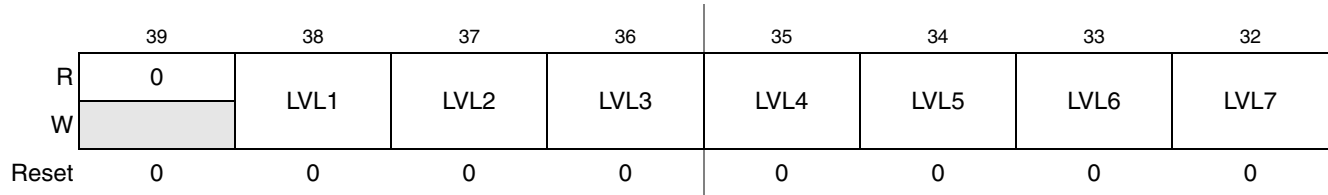
Figure 8-2. INTC_FRC Register

Table 8-4. INTC_FRC Field Descriptions

Field	Description
39	Reserved, must be cleared.
38 LVL1	Force Level 1 interrupt. 0 Negates the forced level 1 interrupt request. 1 Forces a level 1 interrupt request.
37 LVL2	Force Level 2 interrupt. 0 Negates the forced level 2 interrupt request. 1 Forces a level 2 interrupt request.
36 LVL3	Force Level 3 interrupt. 0 Negates the forced level 3 interrupt request. 1 Forces a level 3 interrupt request.
35 LVL4	Force Level 4 interrupt. 0 Negates the forced level 4 interrupt request. 1 Forces a level 4 interrupt request.
34 LVL5	Force Level 5 interrupt. 0 Negates the forced level 5 interrupt request. 1 Forces a level 5 interrupt request.
33 LVL6	Force Level 6 interrupt. 0 Negates the forced level 6 interrupt request. 1 Forces a level 6 interrupt request.
32 LVL7	Force Level 7 interrupt. 0 Negates the forced level 7 interrupt request. 1 Forces a level 7 interrupt request.

8.3.2.2   INTC Programmable Level 6, Priority {7,6} Registers (INTC_PL6P{7,6})

The two level seven interrupt requests (the IRQ package pin and the low-voltage detection interrupt) cannot have their levels reassigned. However, any of the remaining peripheral interrupt requests can be reassigned as the highest priority maskable requests using these two registers. The vector number associated with the interrupt requests is not changed. Rather, only the interrupt request's level and priority are altered, based on the contents of the INTC_PL6P{7,6} registers.

NOTE

The requests associated with the INTC_FRC register have a fixed level and priority that cannot be altered.

The INTC_PL6P7 register specifies the highest-priority, maskable interrupt request, which is defined as the level six, priority seven request. The INTC_PL6P6 register specifies the second-highest-priority, maskable interrupt request defined as the level six, priority six request. Reset clears both registers, disabling any request re-mapping.

For an example of the use of these registers, see Section 8.6.2, "Using INTC_PL6P{7,6} Registers".

Offset: CF1_INTC_BASE + 0x18 (INTC_PL6P7)   Access: Read/Write
        CF1_INTC_BASE + 0x19 (INTC_PL6P6)

	7	6	5	4	3	2	1	0
R	0	0	0			REQN		
W								
Reset	0	0	0	0	0	0	0	0
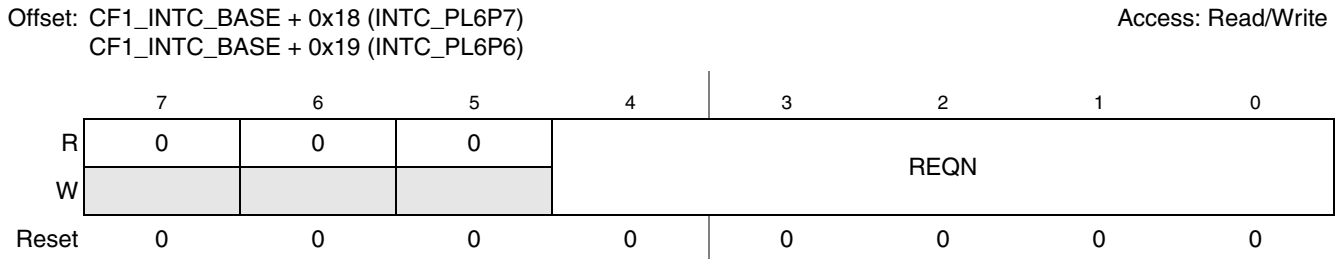
Figure 8-3. INTC_PL6P{7,6} Registers

Table 8-5. INTC_PL6P{7,6} Field Descriptions

Field	Description
7–5	Reserved, must be cleared.
4–0 REQN	Request number. Defines the peripheral IRQ number to be remapped as the level 6, priority 7 (for INTC_PL6P7) request (priority 6 for INTC_PL6P6). The value must be in the 2–29 (0x2–0x1D) range; all other values are ignored.

8.3.2.3 INTC Wake-up Control Register (INTC_WCR)

The interrupt controller provides a combinatorial logic path to generate a special wake-up signal to exit from the wait or stop modes. The INTC_WCR register defines wake-up condition for interrupt recognition during wait and stop modes. This mode of operation works as follows:

1. Write to the INTC_WCR to enable this operation (INTC_WCR[ENB]) and define the interrupt mask level needed to force the core to exit the wait or stop mode (INTC_WCR[MASK]). The maximum value of INTC_WCR[MASK] is 0x6 (0b110).

2. Execute a stop instruction to place the processor into wait or stop mode.

3. After the processor is stopped, the interrupt controller enables special logic that evaluates the incoming interrupt sources in a purely combinatorial path; no clocked storage elements are involved.

4. If an active interrupt request is asserted and the resulting interrupt level is greater than the mask value contained in INTC_WCR[MASK], the interrupt controller asserts the wake-up output signal. This signal is routed to the clock generation logic to exit the low-power mode and resume processing.

Typically, the interrupt mask level loaded into the processor's status register field (SR[I]) during the execution of the STOP instruction matches the INTC_WCR[MASK] value.

The interrupt controller's wake-up signal is defined as:

```
wake-up = INTC_WCR[ENB] & (level of any asserted_int_request > INTC_WCR[MASK])
```

Reset state of the INTC_WCR is disabled, so this register must be written to enable the wake-up condition before the core executes any STOP instructions.

Offset: CF1_INTC_BASE + 0x1B (INTC_WCR)   Access: Read/Write

	7	6	5	4	3	2	1	0
R	ENB	0	0	0	0		MASK	
W								
Reset	0	0	0	0	0	0	0	0
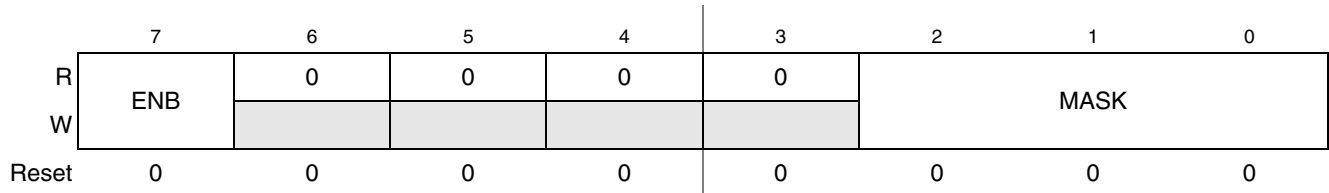
Figure 8-4. INTC_WCR Register

Table 8-6. INTC_WCR Field Descriptions

Field	Description
7 ENB	Enable. 0 Wake-up signal disabled. 1 Enables the assertion of the combinational wake-up signal to the clock generation logic.
6–3	Reserved, must be cleared.
2–0 MASK	Interrupt mask level. Defines the interrupt mask level during wait or stop mode and is enforced by the hardware to be within the range 0–6. If INTC_WCR[ENB] is set, after an interrupt request of a level higher than MASK is asserted, the wake-up signal to the clock generation logic is asserted.

8.3.2.4   INTC Set Interrupt Force Register (INTC_SFRC)

The INTC_SFRC register provides a simple memory-mapped mechanism to set a given bit in the INTC_FRC register to assert a specific level interrupt request. The data value written causes the appropriate bit in the INTC_FRC register to be set. Attempted reads of this register generate an error termination.

This register is provided so interrupt service routines can generate a forced interrupt request without the need to perform a read-modify-write sequence on the INTC_FRC register.

Offset: CF1_INTC_BASE + 0x1E (INTC_SFRC)   Access: Write-only

	7	6	5	4	3	2	1	0
R								
W	0	0			SET			
Reset	0	0	0	0	0	0	0	0
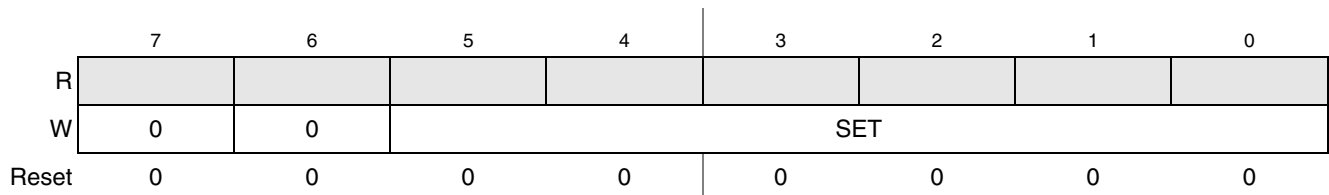
Figure 8-5. INTC_SFRC Register

Table 8-7. INTC_SFRC Field Descriptions

Field	Description
7–6	Reserved, must be cleared.
5–0 SET	For data values within the 32–38 range, the corresponding bit in the INTC_FRC register is set, as defined below. 0x20 Bit 32, INTC_FRC[LVL7] is set. 0x21 Bit 33, INTC_FRC[LVL6] is set. 0x22 Bit 34, INTC_FRC[LVL5] is set. 0x23 Bit 35, INTC_FRC[LVL4] is set. 0x24 Bit 36, INTC_FRC[LVL3] is set. 0x25 Bit 37, INTC_FRC[LVL2] is set. 0x26 Bit 38, INTC_FRC[LVL1] is set. **Note:** Data values outside this range do not affect the INTC_FRC register. It is recommended the data values be restricted to the 0x20–0x26 (32–38) range to insure compatibility with future devices.

8.3.2.5   INTC Clear Interrupt Force Register (INTC_CFRC)

The INTC_CFRC register provides a simple memory-mapped mechanism to clear a given bit in the INTC_FRC register to negate a specific level interrupt request. The data value on the register write causes the appropriate bit in the INTC_FRC register to be cleared. Attempted reads of this register generate an error termination.

This register is provided so interrupt service routines can negate a forced interrupt request without the need to perform a read-modify-write sequence on the INTC_FRC register.

Offset: CF1_INTC_BASE + 0x1F (INTC_CFRC)   Access: Write-only

	7	6	5	4	3	2	1	0
R								
W	0	0			CLR			
Reset	0	0	0	0	0	0	0	0

Figure 8-6. INTC_CFRC Register

Table 8-8. INTC_CFRC Field Descriptions

Field	Description
7–6	Reserved, must be cleared.
5–0 CLR	For data values within the 32–38 range, the corresponding bit in the INTC_FRC register is cleared, as defined below. 0x20 Bit 32, INTC_FRC[LVL7] is cleared. 0x21 Bit 33, INTC_FRC[LVL6] is cleared. 0x22 Bit 34, INTC_FRC[LVL5] is cleared. 0x23 Bit 35, INTC_FRC[LVL4] is cleared. 0x24 Bit 36, INTC_FRC[LVL3] is cleared. 0x25 Bit 37, INTC_FRC[LVL2] is cleared. 0x26 Bit 38, INTC_FRC[LVL1] is cleared. **Note:** Data values outside this range do not affect the INTC_FRC register. It is recommended the data values be restricted to the 0x20–0x26 (32–38) range to insure compatibility with future devices.

8.3.2.6   INTC Software and Level-*n* IACK Registers (*n* = 1,2,3,...,7)

The eight read-only interrupt acknowledge (IACK) registers can be explicitly addressed via memory-mapped accesses or implicitly addressed via a processor-generated interrupt acknowledge cycle during exception processing when CPUCR[IAE] is set. In either case, the interrupt controller's actions are very similar.

First, consider an IACK cycle to a specific level, a level-n IACK. When this type of IACK arrives in the interrupt controller, the controller examines all currently-active level-n interrupt requests, determines the highest priority within the level, and then responds with the unique vector number corresponding to that specific interrupt source. The vector number is supplied as the data for the byte-sized IACK read cycle.

If there is no active interrupt source at the time of the level-*n* IACK, a special spurious interrupt vector (vector number 24 (0x18)) is returned. It is the responsibility of the service routine to manage this error situation.

This protocol implies the interrupting peripheral is not accessed during the acknowledge cycle because the interrupt controller completely services the acknowledge. This means the interrupt source must be explicitly disabled in the peripheral device by the interrupt service routine. This approach provides unique vector capability for all interrupt requests, regardless of the complexity of the peripheral device.

Second, the interrupt controller also supports the concept of a software IACK. This is the ability to query the interrupt controller near the end of an interrupt service routine (after the current interrupt request has been negated) to determine if there are any pending (but currently masked) interrupt requests. If the response to the software IACK's byte operand read is non-zero, the service routine uses the returned value as the vector number of the highest pending interrupt request and passes control to the appropriate new handler. If the returned value is zero, there is no pending interrupt request.

This process avoids the overhead of a context restore and RTE instruction execution, followed immediately by another interrupt exception and context save. In system environments with high rates of interrupt activity, this mechanism can noticeably improve overall performance. For additional details on software IACKs, see Section 8.6.3, "More on Software IACKs."

Offset: CF1_INTC_BASE + 0x20 (INTC_SWIACK)   Access: Read-only
         CF1_INTC_BASE + 0x20 + (4×*n*) (INTC_LVL*n*lACK)

	7	6	5	4	3	2	1	0
R	0	VECN						
W								
SWIACK Reset	0	0	0	0	0	0	0	0
LVL*n*lACK Reset	0	0	0	1	1	0	0	0

Figure 8-7. INTC_SWIACK, INTC_LVLnIACK Registers

Table 8-9. INTC_SWIACK, INTC_LVLnIACK Field Descriptions

Field	Description
7	Reserved, must be cleared.
6–0 VECN	Vector number. Indicates the appropriate vector number. For the SWIACK register, it is the highest-level, highest-priority request currently being asserted in the CF1_INTC module. If there are no pending requests, VECN is zero. For the LVL*n*IACK register, it is the highest priority request within the specified level-*n*. If there are no pending requests within the level, VECN is 0x18 (24) to signal a spurious interrupt.

8.3.3 Interrupt Request Level and Priority Assignments

This section provides two views of the interrupt request assignment: a two-dimensional view of levels and priorities within the level (Table 8-11) and a tabular representation based on request priority (Table 8-12).

The CF1_INTC module implements a sparsely-populated 7×9 matrix of levels (7) and priorities within each level (9). In this representation, the leftmost top cell (level 7, priority 7) is the highest interrupt request while the rightmost lowest cell (level 1, priority 0) is the lowest interrupt request. The following legend is used for this table:

Table 8-10. Legend for Table 8-11

Interrupt Request Source	
Interrupt Source Number	Vector Number

NOTE

For remapped and forced interrupts, the interrupt source number entry indicates the register or register field that enables the corresponding interrupt.

Table 8-11. V1 ColdFire [Level][Priority within Level] Matrix Interrupt Assignments

Level	Priority within Level								
	7	6	5	4	Midpoint	3	2	1	0
7	—	—	—	—	IRQ_pin	Low_voltage	—	—	force_lvl7
					0 | 64	1 | 65			FRC[32] | 96
6	remapped	remapped	TPM1_ch0	TPM1_ch1	—	TPM1_ch2	—	TPM1_ovfl	force_lvl6
	PL6P7 | *	PL6P6 | *	2 | 66	3 | 67		4 | 68		5 | 69	FRC[33] | 97
5	TPM2_ch0	TPM2_ch1	TPM2_ch12	—	—	—	—	TPM2_ovfl	force_lvl5
	6 | 70	7 | 71	8 | 72					9 | 73	FRC[34] | 98
4	SPI2	SPI1	SCI1_err	SCI1_rx	—	SCI1_tx	—	—	force_lvl4
	10 | 74	11 | 75	12 | 76	13 | 77		14 | 78			FRC[35] | 99

Table 8-11. V1 ColdFire [Level][Priority within Level] Matrix Interrupt Assignments (continued)

Level	Priority within Level								
	7	**6**	**5**	**4**	**Midpoint**	**3**	**2**	**1**	**0**
3	IICx[1]	KBIx[2]	ADC	ACMPx[3]	—	—	—	—	force_lvl3
	15 | 79	16 | 80	17 | 81	18 | 82					FRC[36] | 100
2	—	—	SCI2_err	SCI2_rx	—	SCI2_tx	—	RTC	force_lvl2
			19 | 83	20 | 84		21 | 85		22 | 86	FRC[37] | 101
1	TPM3_ch0	TPM3_ch1	TPM3_ch2	TPM3_ch3	—	TPM3_ch4	TPM3_ch5	TPM3_ovfl	force_lvl1
	32 | 87	24 | 88	25 | 89	26 | 90		27 | 91	28 | 92	29 | 93	FRC[38] | 102

[1] There are two I^{2}C modules on-chip. They share a common interrupt vector.

[2] The keyboard features are available on GPIO ports B and D. The two modules share a common interrupt vector.

[3] There are two analog comparator modules on-chip. They share a common interrupt vector.

Table 8-12 presents the same information on interrupt request assignments, but from the highest priority request to the lowest.

Table 8-12. V1 ColdFire Interrupt Assignments

IRQ Source	Level	Priority within Level	Interrupt Source Number	Vector
IRQ_pin	7	mid	0	64
low_voltage	7	3	1	65
force_lvl7	7	0	INTC_FRC[32]	96
remapped_l6p7	6	7	INTC_PL6P7	*
remapped_l6p6	6	6	INTC_PL6P6	*
TPM1_ch0	6	5	2	66
TPM1_ch1	6	4	3	67
TPM1_ch2	6	3	4	68
TPM1_ovfl	6	1	5	69
force_lvl6	6	0	INTC_FRC[33]	97
TPM2_ch0	5	7	6	70
TPM2_ch1	5	6	7	71
TPM2_ch2	5	5	8	72
TPM2_ovfl	5	1	9	73
force_lvl5	5	0	INTC_FRC[34]	98
SPI2	4	7	10	74
SPI1	4	6	11	75
SCI1_err	4	5	12	76

Table 8-12. V1 ColdFire Interrupt Assignments (continued)

IRQ Source	Level	Priority within Level	Interrupt Source Number	Vector
SCI1_rx	4	4	13	77
SCI1_tx	4	3	14	78
force_lvl4	4	0	INTC_FRC[35]	99
IICx[1]	3	7	15	79
KBIx[2]	3	6	16	80
ADC	3	5	17	81
ACMPx[3]	3	4	18	82
force_lvl3	3	0	INTC_FRC[36]	100
SCI2_err	2	5	19	83
SCI2_rx	2	4	20	84
SCI2_tx	2	3	21	85
RTC	2	2	22	86
force_lvl2	2	0	INTC_FRC[37]	101
TPM3_ch0	1	7	23	87
TPM3_ch1	1	6	24	88
TPM3_ch2	1	5	25	89
TPM3_ch3	1	4	26	90
TPM3_ch4	1	3	27	91
TPM3_ch5	1	2	28	92
TPM3_ovfl	1	1	29	93
force_lvl1	1	0	INTC_FRC[38]	102

[1] There are two I^2C modules on-chip. They share a common interrupt vector.

[2] The keyboard features are available on GPIO ports B and D. The two modules share a common interrupt vector.

[3] There are two analog comparator modules on-chip. They share a common interrupt vector.

8.4   Functional Description

The basic operation of the CF1_INTC has been detailed in the preceding sections. This section describes special rules applicable to non-maskable level seven interrupt requests and the module's interfaces.

8.4.1   Handling of Non-Maskable Level 7 Interrupt Requests

Level seven interrupts are treated as non-maskable, edge-sensitive requests while levels one through six are maskable, level-sensitive requests. As a result of this definition, level seven interrupt requests are a

special case. The edge-sensitive nature of these requests means the encoded 3-bit level input from the CF1_INTC to the V1 ColdFire core must change state before the CPU detects an interrupt. A non-maskable interrupt (NMI) is generated each time the encoded interrupt level changes to level seven (regardless of the SR[I] field) and each time the SR[I] mask changes from seven to a lower value while the encoded request level remains at seven.

8.5   Initialization Information

The reset state of the CF1_INTC module enables the default IRQ mappings and clears any software-forced interrupt requests (INTC_FRC is cleared). The wake-up control register (INTC_WCR) is also disabled, so it must be written before the processor executes any stop instructions to properly exit from any wait or stop mode. Immediately after reset, the CF1_INTC begins its cycle-by-cycle evaluation of any asserted interrupt requests and forms the appropriate encoded interrupt level and vector information for the V1 processor core.

8.6   Application Information

This section discusses three application topics: emulation of the HCS08's one level interrupt nesting structure, elevating the priority of two IRQs, and more details on the operation of the software interrupt acknowledge (SWIACK) mechanism.

8.6.1   Emulation of the HCS08's 1-Level IRQ Handling

As noted in Table 8-1, the HCS08 architecture specifies a 1-level IRQ nesting capability. Interrupt masking is controlled by CCR[I], the interrupt mask flag: clearing CCR[I] enables interrupts, while setting CCR[I] disables interrupts. The ColdFire architecture defines seven interrupt levels, controlled by the 3-bit interrupt priority mask field in the status register, SR[I], and the hardware automatically supports nesting of interrupts.

To emulate the HCS08's 1-level IRQ capabilities on V1 ColdFire, only two SR[I] settings are used:

- Writing 0 to SR[I] enables interrupts.
- Writing 7 to SR[I] disables interrupts.

ColdFire treats the two level seven requests (IRQ pin and Low voltage detect) as non-maskable, edge-sensitive interrupts.

ColdFire processors inhibit interrupt sampling during the first instruction of all exception handlers. This allows any handler to effectively disable interrupts, if necessary, by raising the interrupt mask level contained in the status register as the first instruction in the ISR. In addition, the V1 instruction set architecture (ISA_C) includes an instruction (STLDSR) that stores the current interrupt mask level and loads a value into the SR. This instruction is specifically intended for use as the first instruction of an interrupt service routine which services multiple interrupt requests with different interrupt levels. For more details see the *ColdFire Family Programmer's Reference Manual*. A MOVE-to-SR instruction also performs a similar function.

To emulate the HCS08's 1-level IRQ nesting mechanisms, the ColdFire implementation enables interrupts by clearing SR[I] (typically when using RTE to return to a process) and disables interrupts upon entering every interrupt service routine by one of three methods:

1. Execution of STLDSR #0x2700 as the first instruction of an ISR.

2. Execution of MOVE.w #0x2700,SR as the first instruction of an ISR.

3. Static assertion of CPUCR[IME], which forces the processor to load SR[I] with seven automatically upon the occurrence of an interrupt exception. Because this method removes the need to execute multi-cycle instructions of #1 or #2, this approach improves system performance.

8.6.2 Using INTC_PL6P{7,6} Registers

Section 8.3.2.2, "INTC Programmable Level 6, Priority {7,6} Registers (INTC_PL6P{7,6})," describes control registers that provide the ability to dynamically alter the request level and priority of two IRQs. Specifically, these registers provide the ability to reassign two IRQs to be the highest level 6 (maskable) requests. Consider the following example.

Suppose the system operation desires to remap the receive and transmit interrupt requests of a serial communication device (SCI1) as the highest two maskable interrupts. The default assignments for the SCI1 transmit and receive interrupts are:

- sci1_rx = interrupt source 13 = vector 77 = level 4, priority 4
- sci1_tx = interrupt source 14 = vector 78 = level 4, priority 3

To remap these two requests, the INTC_PL6P{7,6} registers are programmed with the desired interrupt source number:

If INTC_PL6P7 equals 13 (0x0D), sci1_rx = interrupt source 13 = vector 77 remapped as level 6, priority 7.

If INTC_PL6P6 equals 14 (0x0E), sci1_tx = interrupt source 14 = vector 78 remapped as level 6, priority 6.

The reset state of the INTC_PL6P{7,6} registers disables any request remapping.

8.6.3 More on Software IACKs

As previously mentioned, the notion of a software IACK refers to the ability to query the interrupt controller near the end of an interrupt service routine (after the current interrupt request has been cleared) to determine if there are any pending (but currently masked) interrupt requests. If the response to the software IACK's byte operand read is non-zero, the service routine uses the value as the vector number of the highest pending interrupt request and passes control to the appropriate new handler. This process avoids the overhead of a context restore and RTE instruction execution, followed immediately by another interrupt exception and context save. In system environments with high rates of interrupt activity, this mechanism can improve overall system performance noticeably.

To illustrate this concept, consider the following ISR code snippet shown in Figure 8-8.

```
                  align   4
                  irqxx_entry:
00588: 4fef fff0  lea     -16(sp),sp          # allocate stack space
0058c: 48d7 0303  movem.l #0x0303,(sp)        # save d0/d1/a0/a1 on stack

                  irqxx_alternate_entry:
00590:
     ....
                  irqxx_swiack:
005c0: 71b8 ffe0  mvz.b   INTC_SWIACK.w,d0    # perform software IACK
005c4: 0c00 0041  cmpi.b  #0x41,d0            # pending IRQ or level 7?
005c8: 6f0a       ble.b   irqxx_exit          # no pending IRQ, then exit
005ca: 91c8       sub.l   a0,a0               # clear a0
005cc: 2270 0c00  move.l  0(a0,d0.l*4),a1     # fetch pointer from xcpt table
005d0: 4ee9 0008  jmp     8(a1)               # goto alternate isr entry point

                  align   4
                  irqxx_exit:
005d4: 4cd7 0303  movem.l (sp),#0x0303        # restore d0/d1/a0/a1
005d8: 4fef 0010  lea     16(sp),sp           # deallocate stack space
005dc: 4e73       rte                         # return from handler
```

Figure 8-8. ISR Code Snippet with SWIACK

This snippet includes the prologue and epilogue for an interrupt service routine as well as code needed to perform software IACK.

At the entry point (`irqxx_entry`), there is a two-instruction prologue to allocate space on the supervisor stack to save the four volatile registers (d0, d1, a0, a1) defined in the ColdFire application binary interface. After these registers have been saved, the ISR continues at the alternate entry point.

The software IACK is performed near the end of the ISR, after the source of the current interrupt request has been negated. First, the appropriate memory-mapped byte location in the interrupt controller is read (PC = 0x5C0). The CF1_INTC module returns the vector number of the highest priority pending request. If no request is pending, zero is returned. The compare instruction is needed to manage a special case involving pending level seven requests. Because the level seven requests are non-maskable, ISR is interrupted to service one of these requests. To avoid any race conditions, this check ignores the two level seven vector numbers (0x40, 0x41). The result is the conditional branch (PC = 0x5C8) is taken if there are no pending requests or if the pending request is a level seven.

If there is a pending non-level seven request, execution continues with a three instruction sequence to calculate and then branch to the appropriate alternate ISR entry point. This sequence assumes the exception vector table is based at address 0x(00)00_0000 and that each ISR uses the same two-instruction prologue shown here. The resulting alternate entry point is a fixed offset (8 bytes) from the normal entry point defined in the exception vector table.

The ISR epilogue includes a three instruction sequence to restore the volatile registers from the stack and return from the interrupt exception.

This example is intentionally simple, but does show how performing the software IACK and passing control to an alternate entry point when there is a pending but masked interrupt request can avoid the execution of the ISR epilogue, another interrupt exception, and the ISR prologue.

Chapter 9
Rapid GPIO (RGPIO)

9.1   Introduction

The Rapid GPIO (RGPIO) module provides a 16-bit general-purpose I/O module directly connected to the processor's high-speed 32-bit local platform bus. This connection to the processor's high-speed platform bus plus support for single-cycle, zero wait-state data transfers allows the RGPIO module to provide improved pin performance when compared to more traditional GPIO modules located on the internal slave peripheral bus.

Many of the pins associated with a device may be used for several different functions. Their primary functions are to provide external interfaces to access off-chip resources. When not used for their primary function, many of the pins may be used as general-purpose digital I/O (GPIO) pins. The definition of the exact pin functions and the affected signals is specific to each device. Every GPIO port, including the RGPIO module, has registers that configure, monitor, and control the port pins.

Figure 9-1 shows the MCF51QE128 series block diagram with the RGPIO highlighted.

Get the latest version from freescale.com
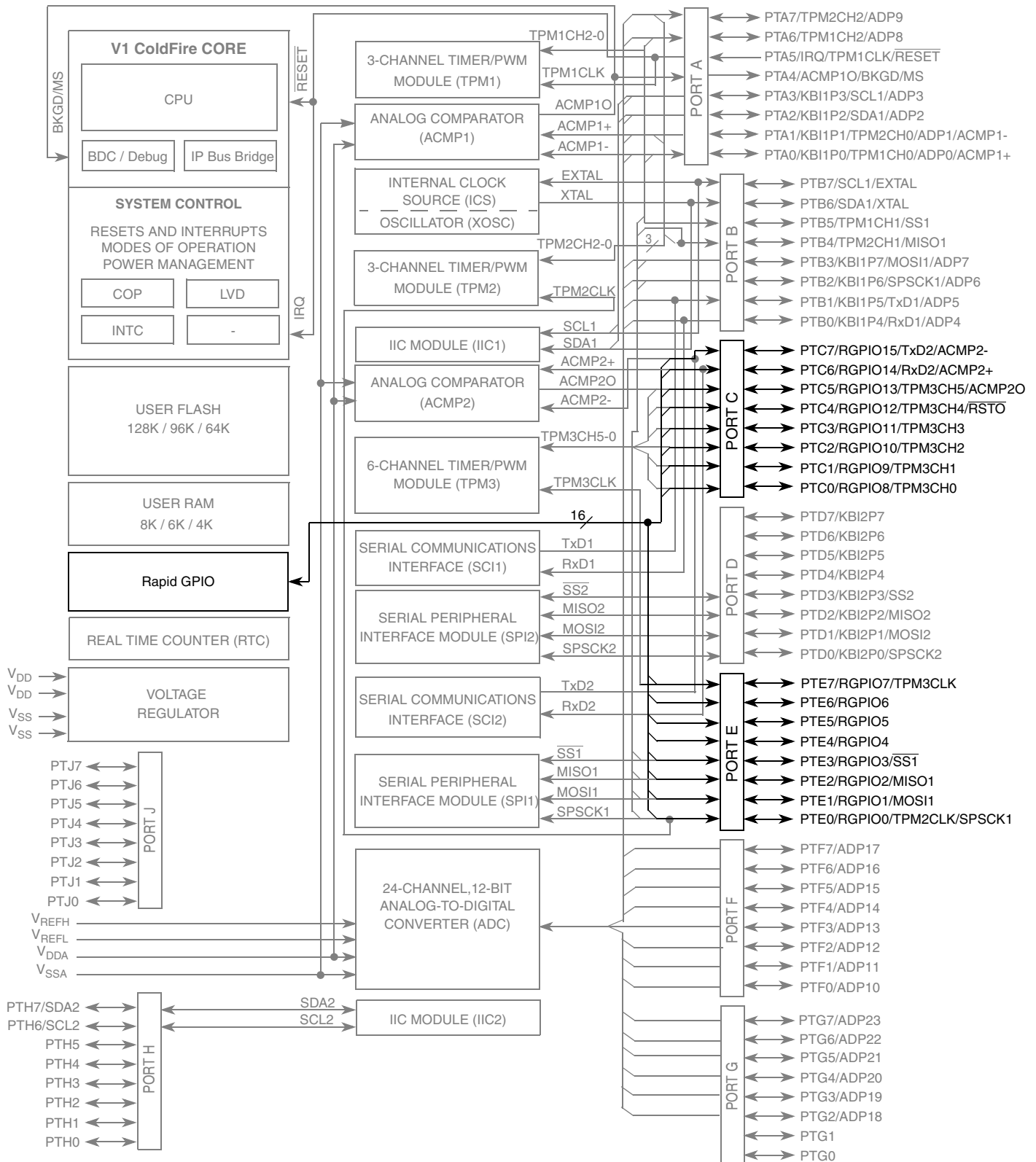
Figure 9-1. MCF51QE128 Series Block Diagram Highlighting RGPIO Block and Pins

9.1.1 Overview

The RGPIO module provides 16-bits of high-speed GPIO functionality, mapped to the processor's platform bus. The key features of this module include:

- 16 bits of high-speed GPIO functionality connected to the processor's local 32-bit platform bus
- Memory-mapped device connected to the ColdFire core's local bus
  - Support for all access sizes: byte, word, and longword
  - All reads and writes complete in a single data phase cycle for zero wait-state response
- Data bits can be accessed directly or via alternate addresses to provide set, clear, and toggle functions
  - Alternate addresses allow set, clear, toggle functions using simple store operations without the need for read-modify-write references
- Unique data direction and pin enable control registers
- Package pin toggle rates typically 1.5–3.5x faster than comparable pin mapped onto peripheral bus

The relative location of the RGPIO module (highlighted in the diagram) within the low-cost Version 1 ColdFire core platform is shown in Figure 9-2.
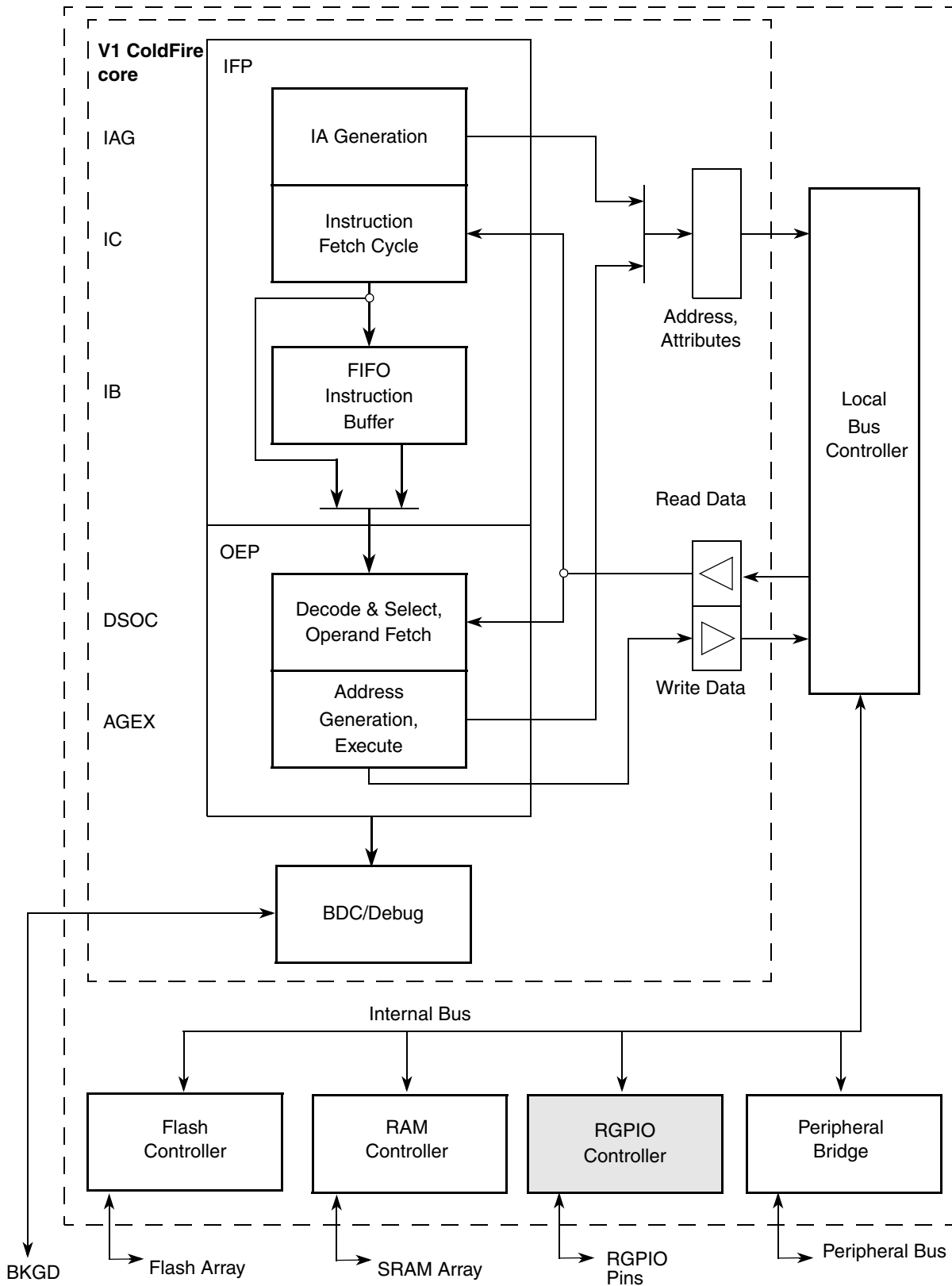
Figure 9-2. V1 ColdFire Block Diagram

A simplified block diagram of the RGPIO module is shown in Figure 9-3. The details of the pin muxing and pad logic are device-specific.
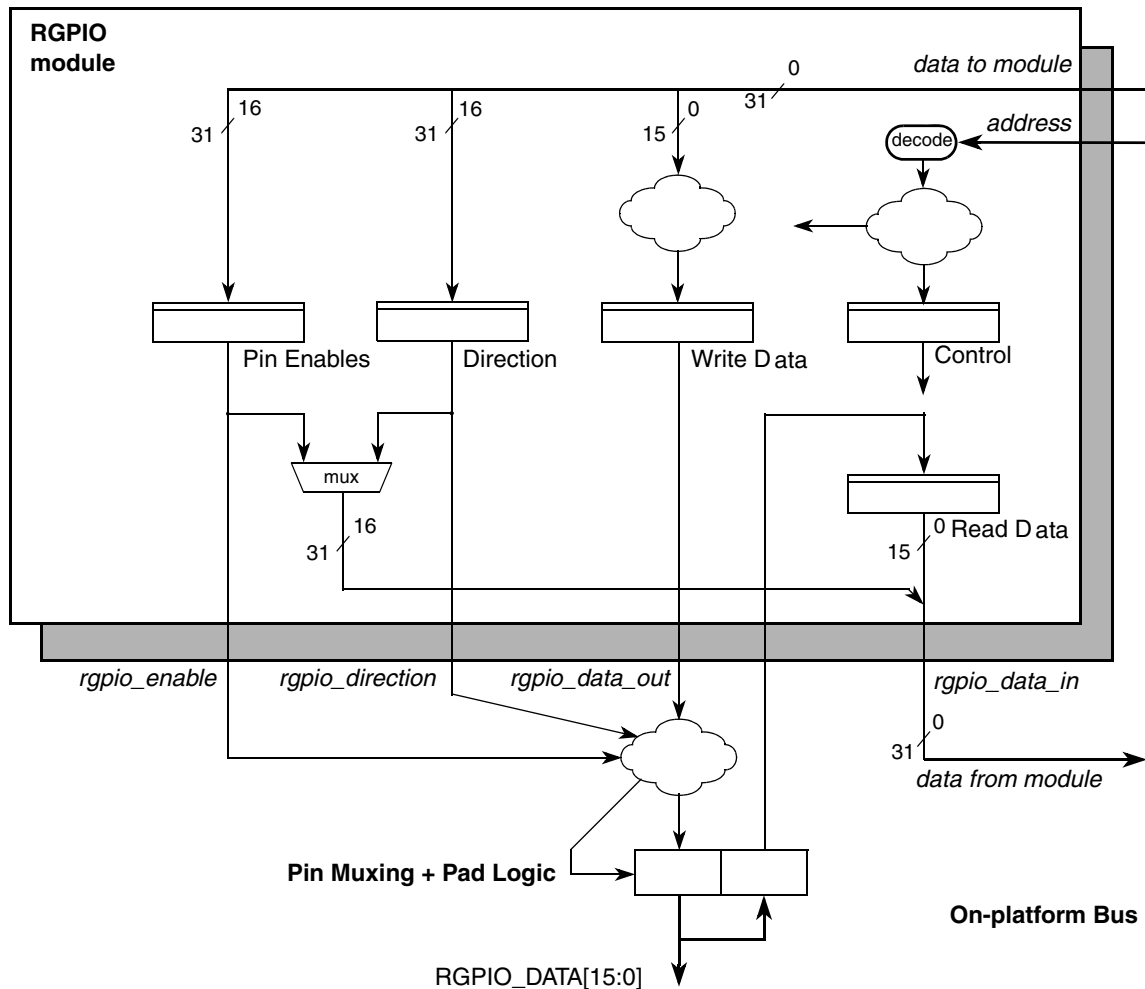
Figure 9-3. RGPIO Block Diagram

9.1.2 Features

The major features of the RGPIO module providing 16 bits of high-speed general-purpose input/output are summarized below:

- Small memory-mapped device connected to the processor's local bus
    - All memory references complete in a single cycle to provide zero wait-state responses
    - Located in processor's high-speed clock domain
- Simple programming model
    - Four 16-bit registers, mapped as three program-visible locations
        - Register for pin enables
        - Register for controlling the pin data direction
        - Register for storing output pin data

- – Register for reading current pin state
- – The two data registers (read, write) are mapped to a single program-visible location
- — Alternate addresses to perform data set, clear, and toggle functions using simple writes
- — Separate read and write programming model views enable simplified driver software
  - – Support for any access size (byte, word, or longword)

9.1.3 Modes of Operation

The RGPIO module does not support any special modes of operation. As a memory-mapped device located on the processor's high-speed local bus, it responds based strictly on memory address and does not consider the operating mode (supervisor, user) of its references.

9.2 External Signal Description

9.2.1 Overview

As shown in Figure 9-3, the RGPIO module's interface to external logic is indirect via the device pin-muxing and pad logic. For a list of the associated RGPIO input/output signals, see Table 9-1.

Table 9-1. RGPIO Module External I/O Signals

Signal Name	Type	Description
RGPIO[15:0]	I/O	RGPIO Data Input/Output

9.2.2 Detailed Signal Descriptions

Table 9-2 provides descriptions of the RGPIO module's input and output signals.

Table 9-2. RGPIO Detailed Signal Descriptions

Signal	I/O	Description		
RGPIO[15:0]	I/O	Data Input/Output. When configured as an input, the state of this signal is reflected in the read data register. When configured as an output, this signal is the output of the write data register.		
		State Meaning	Asserted— Input: Indicates the RGPIO pin was sampled as a logic high at the time of the read. Output: Indicates a properly-enabled RGPIO output pin is to be driven high. Negated— Input: Indicates the RGPIO pin was sampled as a logic low at the time of the read. Output: Indicates a properly-enabled RGPIO output pin is to be driven low.
		Timing	Assertion/Negation— Input: Anytime. The input signal is sampled at the rising-edge of the processor's high-speed clock on the data phase cycle of a read transfer of this register. Output: Occurs at the rising-edge of the processor's high-speed clock on the data phase cycle of a write transfer to this register. This output is asynchronously cleared by system reset.

9.3 Memory Map/Register Definition

The RGPIO module provides a compact 16-byte programming model based at a system memory address of 0x(00)C0_0000 (noted as RGPIO_BASE throughout the chapter). As previously noted, the programming model views are different between reads and writes as this enables simplified software for manipulation of the RGPIO pins. Additionally, the programming model can be referenced using any operand size access (byte, word, longword).

9.3.1 Memory Map

The RGPIO programming model maps are shown in Table 9-3 and Table 9-4.

Table 9-3. RGPIO *Write* Memory Map

Offset Address	Register	Width (bits)	Access	Reset Value	Section/Page
0x0	RGPIO Data Direction Register (RGPIO_DIR)	16	W	0x0000	9.3.2.1/9-200
0x2	RGPIO Write Data Register (RGPIO_DATA)	16	W	0x0000	9.3.2.2/9-200
0x4	RGPIO Pin Enable Register (RGPIO_ENB)	16	W	0x0000	9.3.2.3/9-201
0x6	RGPIO Write Data Clear Register (RGPIO_CLR)	16	W	N/A	9.3.2.4/9-201
0xA	RGPIO Write Data Set Register (RGPIO_SET)	16	W	N/A	9.3.2.5/9-202
0xE	RGPIO Write Data Toggle Register (RGPIO_TOG)	16	W	N/A	9.3.2.6/9-202

Table 9-4. RGPIO *Read* Memory Map

Offset Address	Register	Width (bits)	Access	Reset Value	Section/Page
0x0	RGPIO data direction register (RGPIO_DIR)	16	R	0x0000	9.3.2.1/9-200
0x2	RGPIO write data register (RGPIO_DATA)	16	R	0x0000	9.3.2.2/9-200
0x4	RGPIO pin enable register (RGPIO_ENB)	16	R	0x0000	9.3.2.3/9-201
0x6	RGPIO write data register (RGPIO_DATA)	16	R	0x0000	9.3.2.2/9-200
0x8	RGPIO data direction register (RGPIO_DIR)	16	R	0x0000	9.3.2.1/9-200
0xA	RGPIO write data register (RGPIO_DATA)	16	R	0x0000	9.3.2.2/9-200
0xC	RGPIO data direction register (RGPIO_DIR)	16	R	0x0000	9.3.2.1/9-200
0xE	RGPIO write data register (RGPIO_DATA)	16	R	0x0000	9.3.2.2/9-200

NOTE

Writes to the 2-byte fields at RGPIO_BASE + 0x8 and RGPIO_BASE + 0xC are allowed, but do not affect any program-visible register within the RGPIO module.

9.3.2 Register Descriptions

The RGPIO module provides 16 bits of high-speed general-purpose input/output functionality via a connection to the processor's 32-bit local bus. As a result, the RGPIO programming model is defined with a 32-bit organization. The basic size of each program-visible register is 16 bits, but the programming model may be referenced using byte (8-bit), word (16-bit), or longword (32-bit) accesses. Performance is typically maximized using 32-bit accesses.

9.3.2.1 RGPIO Data Direction (RGPIO_DIR)

The RGPIO_DIR register defines whether a properly-enabled RGPIO pin is configured as an input or output. The RGPIO_DIR register is read/write. At reset, all bits in the RGPIO_DIR are cleared. Setting any bit in the RGPIO_DIR register configures a properly-enabled RGPIO port pin as an output. Clearing any bit in the RGPIO_DIR register configures a properly-enabled RGPIO port pin as an input.

Offset: RGPIO_Base + 0x0 (RGPIO_DIR)   Access: Read/write
       RGPIO_Base + 0x8   Read-only
       RGPIO_Base + 0xC   Read-only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R								DIR								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
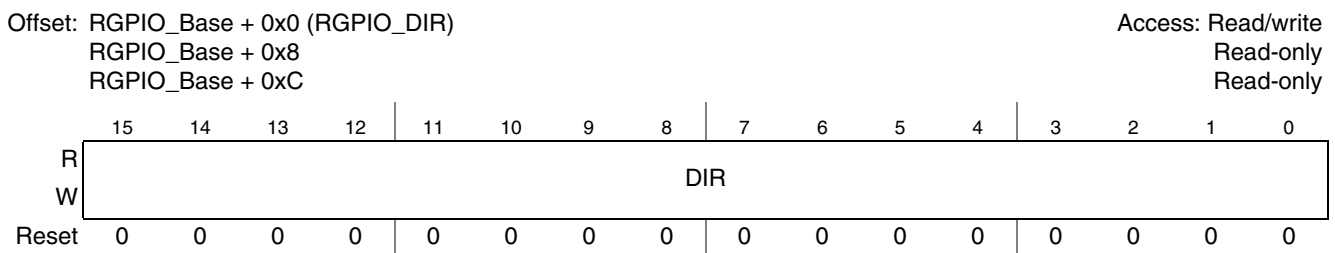
Figure 9-4. RGPIO Data Direction Register (RGPIO_DIR)

Table 9-5. RGPIO_DIR Field Descriptions

Field	Description
15–0 DIR	RGPIO data direction. 0 A properly-enabled RGPIO pin is configured as an input. 1 A properly-enabled RGPIO pin is configured as an output.

9.3.2.2 RGPIO Data (RGPIO_DATA)

The RGPIO_DATA register specifies the write data for a properly-enabled RGPIO output pin or the sampled read data value for a properly-enabled input pin. An attempted read of the RGPIO_DATA register returns undefined data for disabled pins because the data value is dependent on the device-level pin muxing and pad implementation. The RGPIO_DATA register is read/write. At reset, all bits in the RGPIO_DATA registers are cleared.

Reading the RGPIO_DATA register returns the current port values of properly-enabled pins. To set bits in a RGPIO_DATA register, directly set the RGPIO_DATA bits or set the corresponding bits in the RGPIO_SET register. To clear bits in the RGPIO_DATA register, directly clear the RGPIO_DATA bits, or clear the corresponding bits in the RGPIO_CLR register. Setting a bit in the RGPIO_TOG register inverts (toggles) the state of the corresponding bit in the RGPIO_DATA register.

Offset: RGPIO_Base + 0x2 (RGPIO_DATA)   Access: Read/write
          RGPIO_Base + 0x6   Read/Indirect Write
          RGPIO_Base + 0xA   Read/Indirect Write
          RGPIO_Base + 0xE   Read/Indirect Write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W								DATA								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
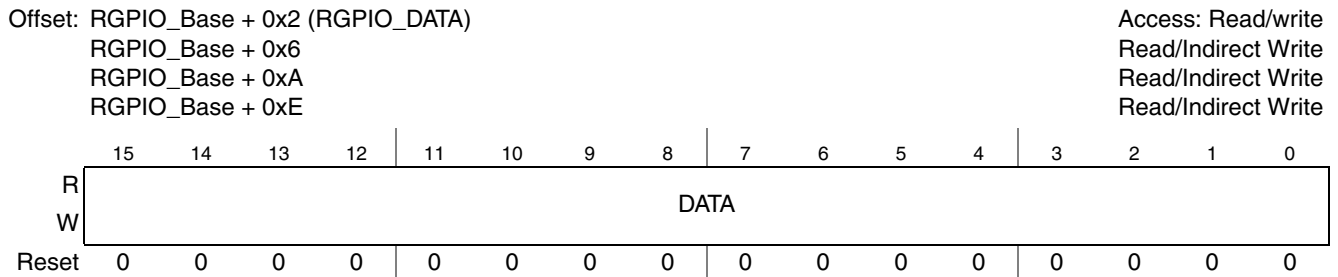
Figure 9-5. RGPIO Data Register (RGPIO_DATA)

Table 9-6. RGPIO_DATA Field Descriptions

Field	Description
15–0 DATA	RGPIO data. 0 A properly-enabled RGPIO output pin is driven with a logic 0, or a properly-enabled RGPIO input pin was read as a logic 0. 1 A properly-enabled RGPIO output pin is driven with a logic 1, or a properly-enabled RGPIO input pin was read as a logic 1.

9.3.2.3   RGPIO Pin Enable (RGPIO_ENB)

The RGPIO_ENB register indicates the corresponding package pin is to be configured as a RGPIO pin instead of the normal GPIO pin mapped onto the peripheral bus.

The RGPIO_ENB register is read/write. At reset, all bits in the RGPIO_ENB are cleared, disabling the RGPIO functionality.

Offset: RGPIO_Base + 0x4 (RGPIO_ENB)   Access: Read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W								ENB								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
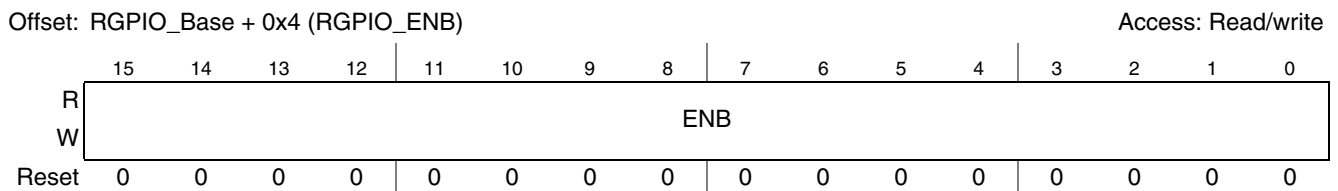
Figure 9-6. RGPIO Enable Register (RGPIO_ENB)

Table 9-7. RGPIO_ENB Field Descriptions

Field	Description
15–0 ENB	RGPIO enable. 0 The corresponding package pin is configured for use as a normal GPIO pin, not a RGPIO. 1 The corresponding package pin is configured for use as a RGPIO pin.

9.3.2.4   RGPIO Clear Data (RGPIO_CLR)

The RGPIO_CLR register provides a mechanism to clear specific bits in the RGPIO_DATA by performing a simple write. Clearing a bit in RGPIO_CLR clears the corresponding bit in the RGPIO_DATA register. Setting it has no effect. The RGPIO_CLR register is write-only; reads of this address return the RGPIO_DATA register.

Offset: RGPIO_Base + 0x6 (RGPIO_CLR)   Access: Write-only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W								CLR								
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 9-7. RGPIO Clear Data Register (RGPIO_CLR)

Table 9-8. RGPIO_CLR Field Descriptions

Field	Description
15–0 CLR	RGPIO clear data. 0 Clears the corresponding bit in the RGPIO_DATA register. 1 No effect.

9.3.2.5   RGPIO Set Data (RGPIO_SET)

The RGPIO_SET register provides a mechanism to set specific bits in the RGPIO_DATA register by performing a simple write. Setting a bit in RGPIO_SET asserts the corresponding bit in the RGPIO_DATA register. Clearing it has no effect. The RGPIO_SET register is write-only; reads of this address return the RGPIO_DATA register.

Offset: RGPIO_Base + 0xA (RGPIO_SET)   Access: Write-only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W								SET								
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 9-8. RGPIO Set Data Register (RGPIO_SET)

Table 9-9. RGPIO_SET Field Descriptions

Field	Description
15–0 SET	RGPIO set data. 0 No effect. 1 Sets the corresponding bit in the RGPIO_DATA register.

9.3.2.6   RGPIO Toggle Data (RGPIO_TOG)

The RGPIO_TOG register provides a mechanism to invert (toggle) specific bits in the RGPIO_DATA register by performing a simple write. Setting a bit in RGPIO_TOG inverts the corresponding bit in the RGPIO_DATA register. Clearing it has no effect. The RGPIO_TOG register is write-only; reads of this address return the RGPIO_DATA register.
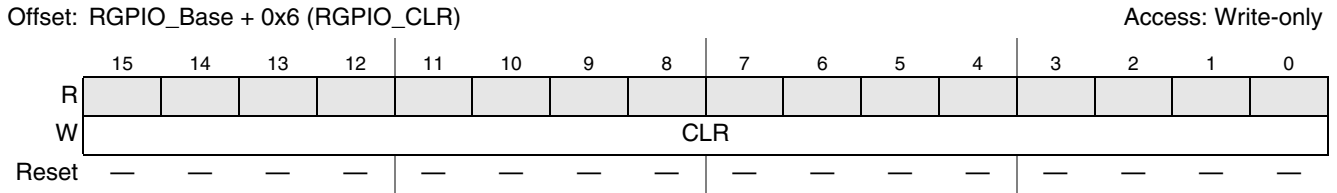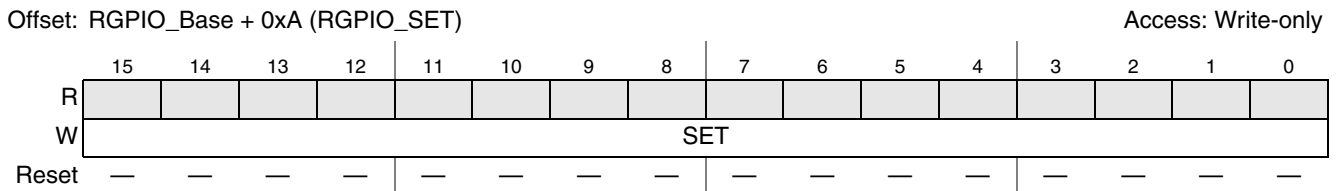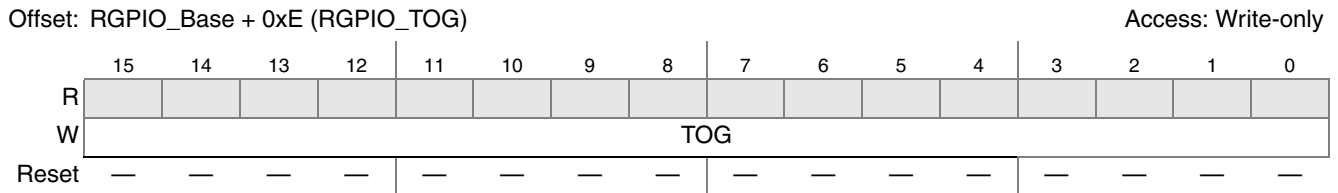
Offset: RGPIO_Base + 0xE (RGPIO_TOG)   Access: Write-only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W								TOG								
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 9-9. RGPIO Toggle Data Register (RGPIO_TOG)

Table 9-10. RGPIO_TOG Field Descriptions

Field	Description
15–0 TOG	RGPIO Toggle Data. 0 No effect. 1 Inverts the corresponding bit in RGPIO_DATA

9.4 Functional Description

The RGPIO module is a relatively-simple design with its behavior controlled by the program-visible registers defined within its programming model.

As shown in Figure 9-2, the RGPIO module is connected to the processor's local two-stage pipelined bus with the stages of the V1 ColdFire core's operand execution pipeline (OEP) mapped directly onto the bus. This structure allows the processor access to the RGPIO module for single-cycle pipelined reads and writes with a zero wait-state response (as viewed in the system bus data phase stage).

9.5 Initialization Information

The reset state of the RGPIO module disables the entire 16-bit data port. Prior to using the RGPIO port, software typically defines the contents of the data register (RGPIO_DATA), configure the pin direction (RGPIO_DIR), and setting the appropriate bits in the pin enable register (RGPIO_ENB).

9.6 Application Information

In this section, the relative performance of the RGPIO output pins for two simple applications is examined. In the first case, the processor executes a loop to toggle an output pin for a specific number of cycles. The speed of the resulting square-wave output is studied. For the second example, the processor transmits a 16-bit message using a 3-pin SPI-like interface with a serial clock, serial chip select and serial data bit. In both cases, the relative speed of the GPIO output is presented as a function of the location of the output bit (RGPIO versus peripheral bus GPIO).

9.6.1 Application 1: Simple Square-Wave Generation

In this example, several different instruction loops are executed, each generating a square-wave output with a 50% duty cycle. For this analysis, the executed code is mapped into the processor's RAM. This configuration is selected to remove any jitter from the output square wave caused by the limitations defined by the two-cycle flash memory accesses and restrictions on the initiation of a flash access. The following instruction loops were studied:

- BCHG_LOOP: In this loop, a bit change instruction was executed using the GPIO data byte as the operand. This instruction performs a read-modify-write operation and inverts the addressed bit. A pulse counter is decremented until the appropriate number of square-wave pulses have been generated.

- SET+CLR_LOOP: For this construct, two store instructions are executed: one to set the GPIO data pin and another to clear it. Single-cycle NOP instructions (the tpf opcode) are included to maintain the 50% duty cycle of the generated square wave. The pulse counter is decremented until the appropriate number of square-wave pulse have been generated.

The square-wave output frequency was measured and the relative performance results are presented in Table 9-11. The relative performance is stated as a fraction of the processor's operating frequency, defined as f MHz. The performance of the BCHG loop operating on a GPIO output is selected as the reference.

Table 9-11. Square-Wave Output Performance

Loop	Peripheral Bus-mapped GPIO			RGPIO		
	Sq-Wave Frequency	Frequency @ CPU f = 50 MHz	Relative Speed	Sq-Wave Frequency	Frequency @ CPU f = 50 MHz	Relative Speed
bchg	$(1/24) \times f$ MHz	2.083 MHz	1.00x	$(1/14) \times f$ MHz	3.571 MHz	1.71x
set+clr (+toggle)	$(1/12) \times f$ MHz	4.167 MHz	2.00x	$(1/8) \times f$ MHz	6.250 MHz	3.00x

NOTE

The square-wave frequency is measured from rising-edge to rising-edge, where the output wave has a 50% duty cycle.

9.6.2   Application 2: 16-bit Message Transmission using SPI Protocol

In this second example, a 16-bit message is transmitted using three programmable output pins. The output pins include a serial clock, an active-high chip select, and the serial data bit. The software is configured to sample the serial data bit at the rising-edge of the clock with the data sent in a most-significant to least-significant bit order. The resulting 3-bit output is shown in Figure 9-10.
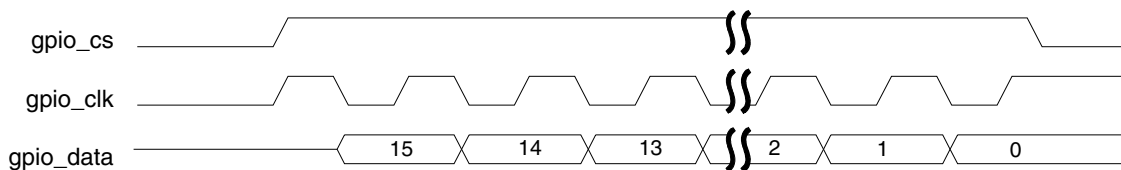
Figure 9-10. GPIO SPI Example Timing Diagram

For this example, the processing of the SPI message is considerably more complex than the generation of a simple square wave of the previous example. The code snippet used to extract the data bit from the message and build the required GPIO data register writes is shown in Figure 9-11.

```
# subtest: send a 16-bit message via a SPI interface using a RGPIO

                # the SPI protocol uses a 3-bit value: clock, chip-select, data
                # the data is centered around the rising-edge of the clock
```

```
                              align   16
                      send_16b_spi_message_rgpio:
00510: 4fef fff4              lea     -12(%sp),%sp         # allocate stack space
00514: 48d7 008c              movm.l  &0x8c,(%sp)          # save d2,d3,d7
00518: 3439 0080 0582         mov.w   RAM_BASE+message2,%d2  # get 16-bit message
0051e: 760f                   movq.l  &15,%d3              # static shift count
00520: 7e10                   movq.l  &16,%d7              # message bit length
00522: 207c 00c0 0003         mov.l   &RGPIO_DATA+1,%a0    # pointer to low-order data byte
00528: 203c 0000 ffff         mov.l   &0xffff,%d0          # data value for _ENB and _DIR regs
0052e: 3140 fffd              mov.w   %d0,-3(%a0)          # set RGPIO_DIR register
00532: 3140 0001              mov.w   %d0,1(%a0)           # set RGPIO_ENB register

00536: 223c 0001 0000         mov.l   &0x10000,%d1         # d1[17:16] = {clk, cs}
0053c: 2001                   mov.l   %d1,%d0              # copy into temp reg
0053e: e6a8                   lsr.l   %d3,%d0              # align in d0[2:0]
00540: 5880                   addq.l  &4,%d0               # set clk = 1
00542: 1080                   mov.b   %d0,(%a0)            # initialize data
00544: 6002                   bra.b   L%1
                              align   4

                      L%1:
00548: 3202                   mov.w   %d2,%d1              # d1[17:15] = {clk, cs, data}
0054a: 2001                   mov.l   %d1,%d0              # copy into temp reg
0054c: e6a8                   lsr.l   %d3,%d0              # align in d0[2:0]
0054e: 1080                   mov.b   %d0,(%a0)            # transmit data with clk = 0
00550: 5880                   addq.l  &4,%d0               # force clk = 1
00552: e38a                   lsl.l   &1,%d2               # d2[15] = new message data bit
00554: 51fc                   tpf                          # preserve 50% duty cycle
00556: 51fc                   tpf
00558: 51fc                   tpf
0055a: 51fc                   tpf
0055c: 1080                   mov.b   %d0,(%a0)            # transmit data with clk = 1
0055e: 5387                   subq.l  &1,%d7               # decrement loop counter
00560: 66e6                   bne.b   L%1

00562: c0bc 0000 fff5         and.l   &0xfff5,%d0          # negate chip-select
00568: 1080                   mov.b   %d0,(%a0)            # update gpio

0056a: 4cd7 008c              movm.l  (%sp),&0x8c          # restore d2,d3,d7
0056e: 4fef 000c              lea     12(%sp),%sp          # deallocate stack space
00572: 4e75                   rts
```

Figure 9-11. GPIO SPI Code Example

The resulting SPI performance, as measured in the effective Mbps transmission rate for the 16-bit message, is shown in Table 9-12.

Table 9-12. Emulated SPI Performance using GPIO Outputs

Peripheral Bus-mapped GPIO		RGPIO	
SPI Speed @ CPU f = 50 MHz	Relative Speed	SPI Speed @ CPU f = 50 MHz	Relative Speed
2.063 Mbps	1.00x	3.809 Mbps	1.29x

Chapter 10
Analog Comparator 3V (ACMPVLPV1)

10.1   Introduction

MCF51QE128 Series MCUs have two independent analog comparators (ACMPs), named ACMP1 and ACMP2.

The analog comparator module (ACMP) provides a circuit for comparing two analog input voltages or for comparing one analog input voltage to an internal reference voltage. The comparator circuit is designed to operate across the full range of the supply voltage (rail-to-rail operation).

Figure 10-1 shows the MCF51QE128 Series block diagram with the ACMP highlighted.

NOTE

Ignore any references to stop1 low-power mode in this chapter, because the MCF51QE128 device does not support it.

10.1.1   ACMP Configuration Information

When using the bandgap reference voltage for input to ACMP1+ and/or ACMP2+, enable the bandgap buffer by setting SPMS[BGBE]. For value of bandgap voltage reference, see the data sheet.

10.1.2   ACMP/TPM Configuration Information

The ACMP modules can be configured to connect the output of the analog comparator to a TPM input capture channel 0 by setting the corresponding ACICx bit in SOPT2. With ACICx set, the TPMxCH0 pin is not available externally regardless of the configuration of the TPMx module.

The ACMP1 output can be connected to TPM1CH0. The ACMP2 output can be connected to TPM2CH0.

10.1.3   ACMP Clock Gating

The bus clock to both of the ACMPs can be gated on and off using the SCGC2[ACMP] bit. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the ACMP bit can be cleared to disable the clock to this module when not in use. See Section 5.6, "Peripheral Clock Gating," for details.

10.1.4   Interrupt Vectors

ACMP1 and ACMP2 share a single interrupt vector. When interrupts are enabled for both ACMPs, the ACF bit in ACMP1SC and ACMP2SC must be polled to determine which ACMP caused the interrupt. See Chapter 8, "Interrupt Controller (CF1_INTC)," for the ACMP interrupt vector assignment.
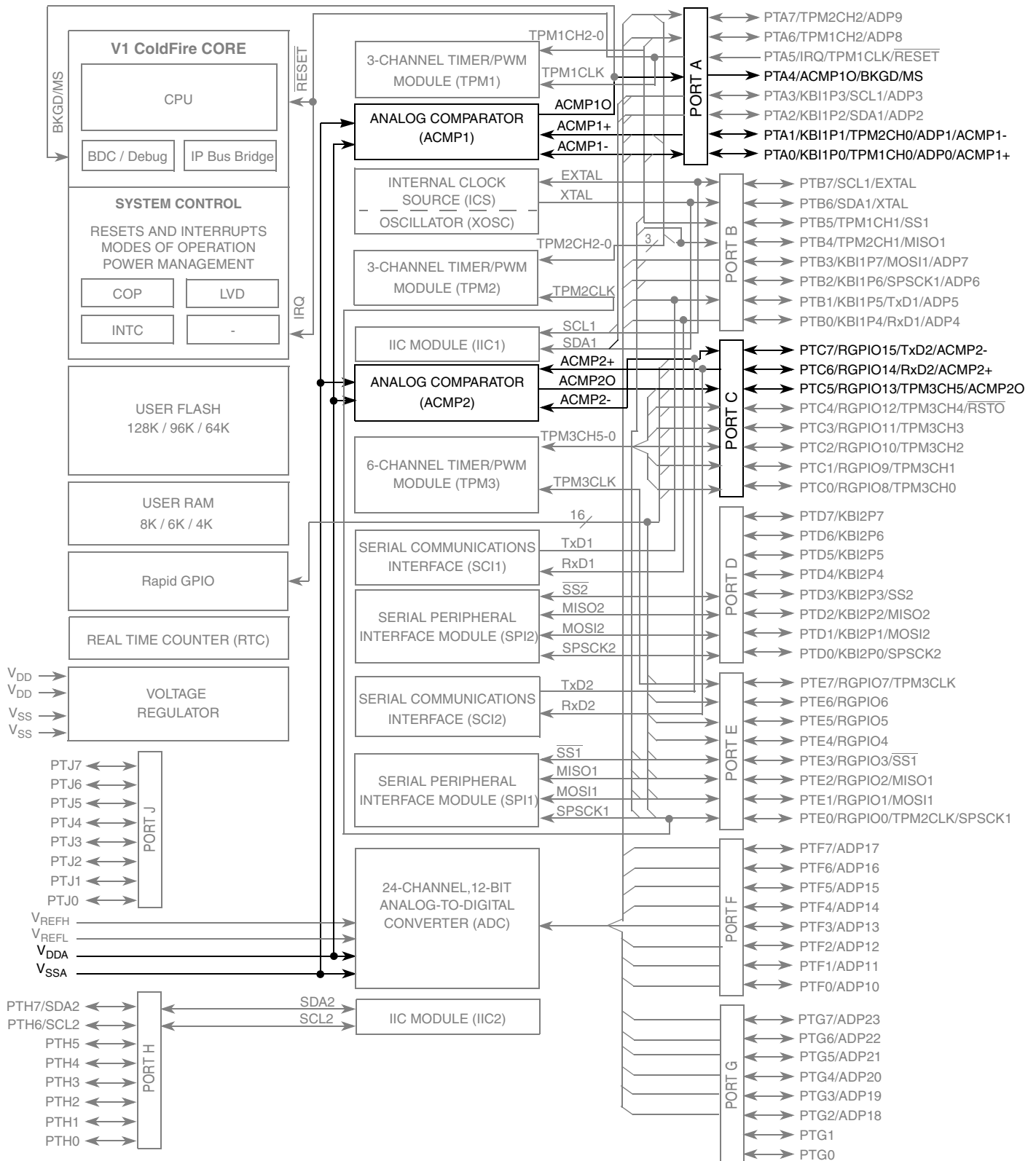
Figure 10-1. MCF51QE128 Series Block Diagram Highlighting ACMP Block and Pins

Get the latest version from freescale.com

10.1.5   Features

The ACMP has the following features:

- Full rail-to-rail supply operation

- Less than 40 mV of input offset

- Less than 15 mV of hysteresis

- Selectable interrupt on rising edge, falling edge, or either rising or falling edges of comparator output

- Option to compare to fixed internal bandgap reference voltage

10.1.6   Modes of Operation

10.1.6.1   Wait Mode Operation

During wait mode the ACMP, if enabled, continues to operate normally. Also, if enabled, the interrupt can wake the MCU.

10.1.6.2   Stop3 Mode Operation

If enabled, the ACMP continues to operate in stop3 mode and compare operation remains active. If ACOPE is enabled, comparator output operates in the normal operating mode and comparator output is placed onto the external pin. The MCU is brought out of stop when a compare event occurs and ACIE is enabled; ACF flag sets accordingly.

If stop is exited with a reset, the ACMP will be put into its reset state.

10.1.6.3   Stop2 Mode Operation

During stop2 mode, the ACMP module is fully powered down. Upon wakeup from stop2 mode, the ACMP module is in the reset state.

10.1.6.4   Active Background Mode Operation

When the microcontroller is in active background mode, the ACMP continues to operate normally.

10.1.7   Block Diagram

The block diagram for the ACMP module follows.
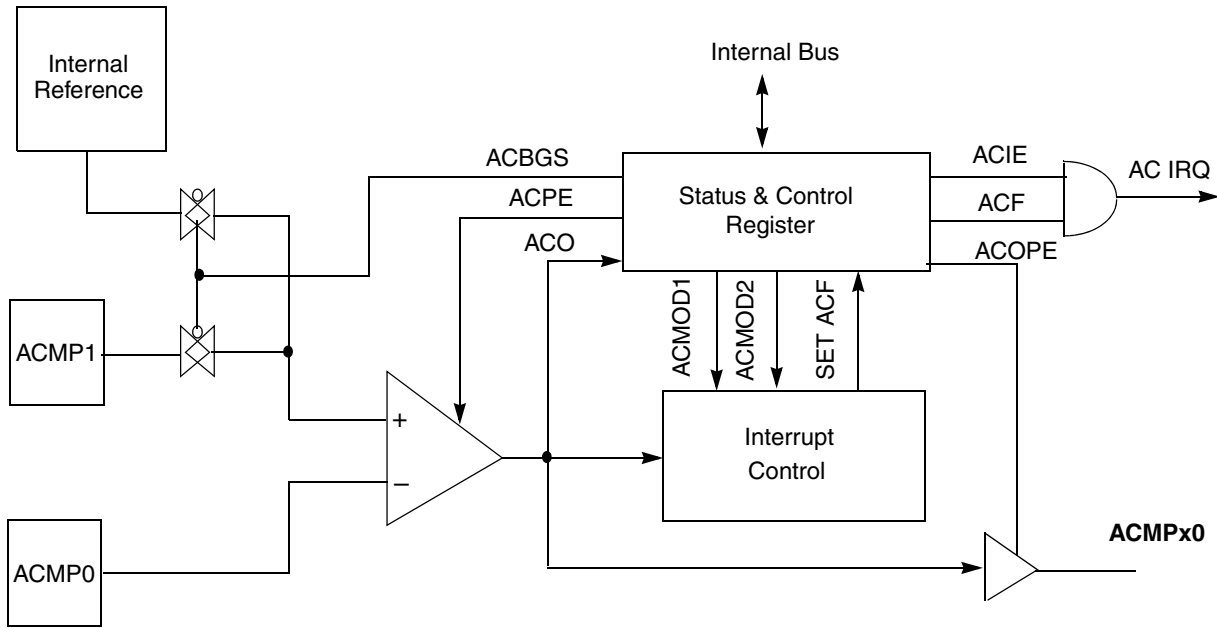
Figure 10-2. Analog Comparator Module Block Diagram

10.2   External Signal Description

The ACMP has two analog input pins: ACMP0 and ACMP1. Each of these pins can accept an input voltage that varies across the full operating voltage range of the MCU. If the module is not enabled, each of these pins can be used as digital inputs or outputs. Consult the specific MCU documentation to determine what functions are shared with these analog inputs. As shown in the block diagram, the ACMP1 pin is connected to the comparator non-inverting input if ACBGS is equal to logic zero, and the ACMP0 pin is connected to the inverting input of the comparator.

10.3   Register Definition

10.3.1   Status and Control Register (ACMPxSC)

	7	6	5	4	3	2	1	0
R	ACME	ACBGS	ACF	ACIE	ACO	ACOPE	ACMOD1	ACMOD0
W								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented
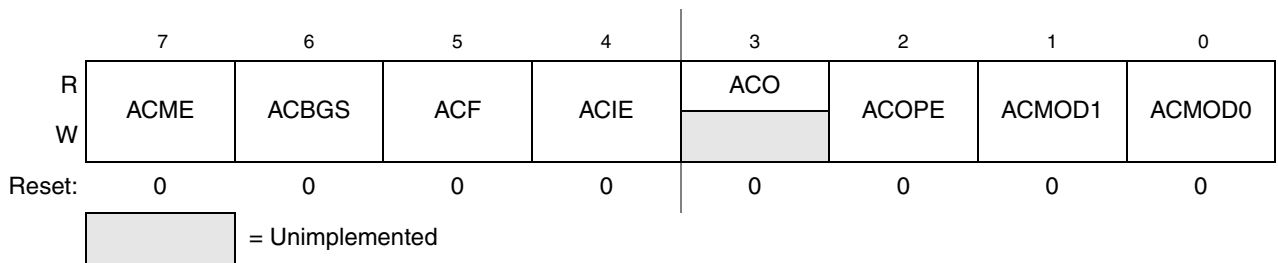
Figure 10-3. ACMP Status and Control Register (ACMPxSC)

Table 10-1. ACMPxSC Field Descriptions

Field	Description
7 ACME	**Analog Comparator Module Enable** — The ACME bit enables the ACMP module. When the module is not enabled, it remains in a low power state. 0 Analog Comparator disabled. 1 Analog Comparator enabled.
6 ACBGS	**Analog Comparator Bandgap Select** — The ACBGS bit selects the internal bandgap as the comparator reference. 0 External pin ACMP1 selected as comparator non-inverting input. 1 Internal bandgap reference selected as comparator non-inverting input.
5 ACF	**Analog Comparator Flag** — The ACF bit is set when a compare event occurs. Compare events are defined by the ACMOD0 and ACMOD1 bits. The ACF bit is cleared by writing a logic one to the bit. 0 Compare event has not occurred. 1 Compare event has occurred.
4 ACIE	**Analog Comparator Interrupt Enable** — The ACIE bit enables the interrupt from the ACM. When this bit is set, an interrupt is asserted when the ACF bit is set. 0 Interrupt disabled. 1 Interrupt enabled.
3 ACO	**Analog Comparator Output** — Reading the ACO bit returns the current value of the analog comparator output. The register bit is reset to zero and reads as logic zero when the ACMP module is disabled (ACME = 0).
2 ACOPE	**Analog Comparator Output Pin Enable** — ACOPE enables the comparator output to be placed onto the external pin, ACMPx1O. 0 Analog comparator output not available on ACMPx1O. 1 Analog comparator output is driven out on ACMPx1O.
1:0 ACMOD	**Analog Comparator Modes** — The ACMOD1 and ACMOD0 bits select the flag setting mode that controls the type of compare event that sets the ACF bit. 00 Comparator output falling edge. 01 Comparator output rising edge. 10 Comparator output falling edge. 11 Comparator output rising or falling edge.

10.4   Functional Description

The ACMP module can be used to compare:

- Two analog input voltages applied to ACMP0 and ACMP1 or
- An analog input voltage applied to ACMP0 with an internal bandgap reference voltage

The ACBGS bit selects the mode of operation. The comparator output is high when the non-inverting input is greater than the inverting input, and low when the non-inverting input is less than the inverting input. The ACMOD0 and ACMOD1 bits select the condition that cause the ACF bit to be set. The ACF bit can be set on a rising edge of the comparator output, a falling edge of the comparator output, or either a rising or a falling edge (toggle). The comparator output can be read directly through the ACO bit.

10.5   Interrupts

The ACMP module is capable of generating an interrupt on a compare event. The interrupt request is asserted when both the ACIE bit and the ACF bit are set. The interrupt is deasserted by clearing either the

ACIE bit or the ACF bit. The ACIE bit is cleared by writing a logic zero and the ACF bit is cleared by writing a logic one.

Chapter 11
Analog-to-Digital Converter (S08ADC12V1)

11.1 Introduction

The 12-bit analog-to-digital converter (ADC) is a successive approximation ADC designed for operation within an integrated microcontroller system-on-chip.

Figure 11-1 shows the MCF51QE128 Series with the ADC module and pins highlighted.

NOTE
Ignore any references to stop1 low-power mode in this chapter, because the MCF51QE128 device does not support it.

11.1.1 ADC Clock Gating

The bus clock to the ADC can be gated on and off using the SCGC1[ADC] bit. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the ADC bit can be cleared to disable the clock to this module when not in use. See Section 5.6, "Peripheral Clock Gating," for details.

Get the latest version from freescale.com
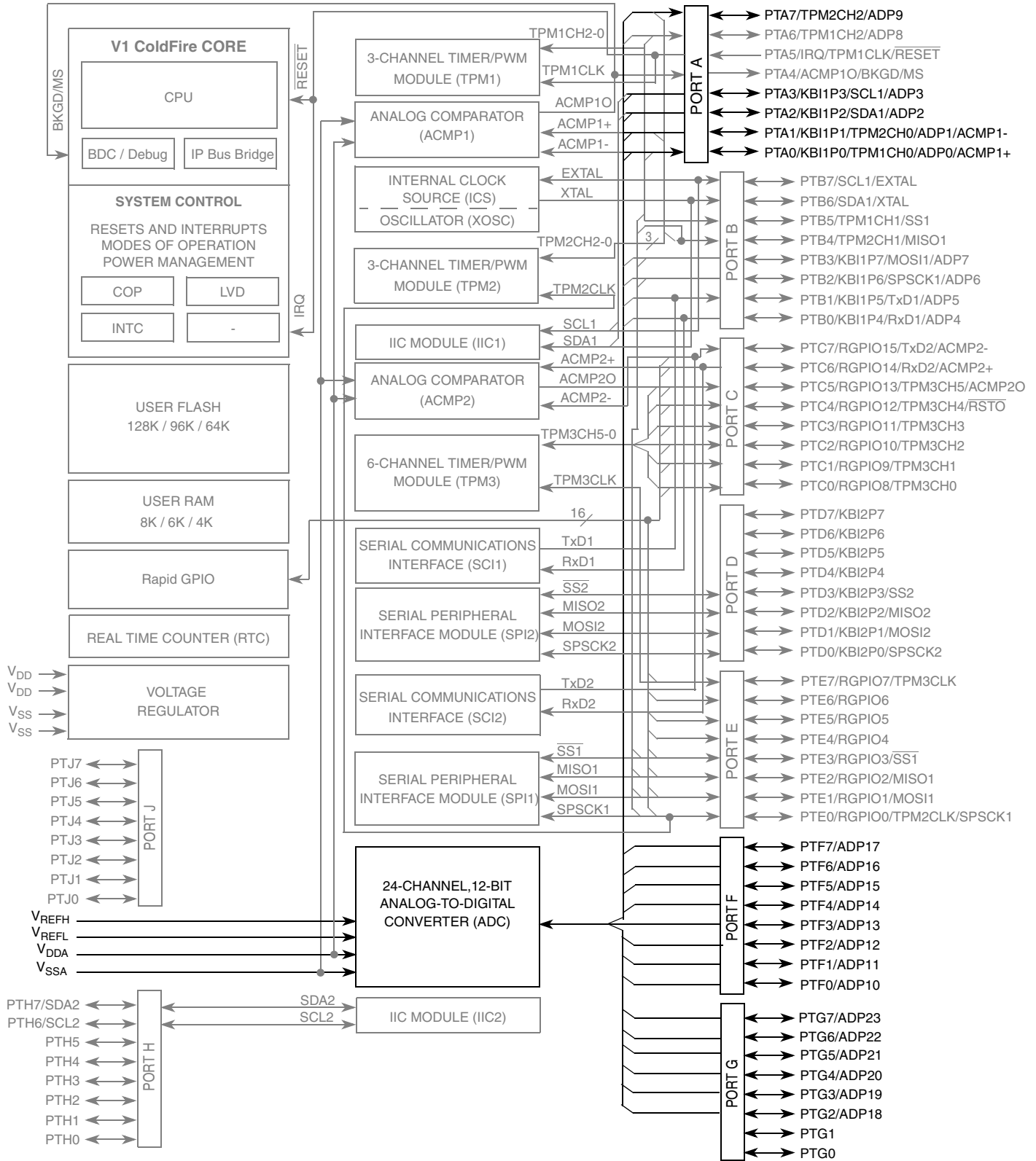
Figure 11-1. MCF51QE128 Series Block Diagram Highlighting ADC Block and Pins

11.1.2   Module Configurations

This section provides device-specific information for configuring the ADC on the MCF51QE128 Series.

11.1.2.1   Channel Assignments

The ADC channel assignments for the MCF51QE128 Series devices are shown in Table 11-1. Reserved channels convert to an unknown value.

Table 11-1. ADC Channel Assignment

ADCH	Channel	Input	Pin Control	ADCH	Channel	Input	Pin Control
00000	AD0	PTA0/ADP0	ADPC0	10000	AD16	PTF6/ADP16	N/A
00001	AD1	PTA1/ADP1	ADPC1	10001	AD17	PTF7/ADP17	N/A
00010	AD2	PTA2/ADP2	ADPC2	10010	AD18	PTG2/ADP18	N/A
00011	AD3	PTA3/ADP3	ADPC3	10011	AD19	PTG3/ADP19	N/A
00100	AD4	PTB0/ADP4	ADPC4	10100	AD20	PTG4/ADP20	N/A
00101	AD5	PTB1/ADP5	ADPC5	10101	AD21	PTG5/ADP21	N/A
00110	AD6	PTB2/ADP6	ADPC6	10110	AD22	PTG6/ADP22	N/A
00111	AD7	PTB3/ADP7	ADPC7	10111	AD23	PTG7/ADP23	N/A
01000	AD8	PTA6/ADP8	N/A	11000	AD24	Reserved	N/A
01001	AD9	PTA7/ADP9	N/A	11001	AD25	Reserved	N/A
01010	AD10	PTF0/ADP10	N/A	11010	AD26	Temperature Sensor[1]	N/A
01011	AD11	PTF1/ADP11	N/A	11011	AD27	Internal Bandgap	N/A
01100	AD12	PTF2/ADP12	N/A	11100	—	Reserved	N/A
01101	AD13	PTF3/ADP13	N/A	11101	V_{REFH}	V_{DD}	N/A
01110	AD14	PTF4/ADP14	N/A	11110	V_{REFL}	V_{SS}	N/A
01111	AD15	PTF5/ADP15	N/A	11111	Module Disabled	None	N/A

[1] For information, see Section 11.1.2.4, "Temperature Sensor."

NOTE

Selecting the internal bandgap channel requires SPMSC1[BGBE] to be set. See Section 5.7.6, "System Power Management Status and Control 1 Register (SPMSC1)". For the value of bandgap voltage reference, see the data sheet.

11.1.2.2   Alternate Clock

The ADC is capable of performing conversions using the MCU bus clock, the bus clock divided by two, the local asynchronous clock (ADACK) within the module, or the alternate clock (ALTCLK). The ALTCLK on the MCF51QE128 Series is the ICSERCLK. See Chapter 12, "Internal Clock Source (S08ICSV3)," for more information.

11.1.2.3 Hardware Trigger

The ADC may initiate a conversion via software or a hardware trigger. The RTC can be enabled as the hardware trigger for the ADC module by setting ADCSC2[ADTRG]. When enabled, the ADC is triggered each time RTCINT matches RTCMOD. The RTC interrupt does not have to be enabled to trigger the ADC.

The RTC can be configured to cause a hardware trigger in MCU run, wait, and stop3.

11.1.2.4 Temperature Sensor

The ADC module includes a temperature sensor whose output is connected to one of the ADC analog channel inputs. Equation 11-1 provides an approximate transfer function of the temperature sensor.

$$\text{Temp} = 25 - \frac{V_{TEMP} - V_{TEMP25}}{m} \qquad \textit{Eqn. 11-1}$$

where:

— V_{TEMP} is the voltage of the temperature sensor channel at the ambient temperature.

— V_{TEMP25} is the voltage of the temperature sensor channel at 25°C.

— m is the hot or cold voltage versus temperature slope in V/°C.

For temperature calculations, use the V_{TEMP25} and m values in the data sheet.

In application code, read the temperature sensor channel, calculate V_{TEMP}, and compare it to V_{TEMP25}. If V_{TEMP} is greater than V_{TEMP25} the cold slope value is applied in Equation 11-1. If V_{TEMP} is less than V_{TEMP25} the hot slope value is applied in Equation 11-1.

11.1.3 Interrupt Vectors

The ADC module contains a single interrupt source. See Chapter 8, "Interrupt Controller (CF1_INTC)," for the ADC interrupt vector assignment.

11.1.4   Features

Features of the ADC module include:

- Linear successive approximation algorithm with 12-bit resolution
- Up to 28 analog inputs
- Output formatted in 12-, 10-, or 8-bit right-justified unsigned format
- Single or continuous conversion (automatic return to idle after single conversion)
- Configurable sample time and conversion speed/power
- Conversion complete flag and interrupt
- Input clock selectable from up to four sources
- Operation in wait or stop3 modes for lower noise operation
- Asynchronous clock source for lower noise operation
- Selectable asynchronous hardware conversion trigger
- Automatic compare with interrupt for less-than, or greater-than or equal-to, programmable value

11.1.5   Block Diagram

Figure 11-2 provides a block diagram of the ADC module.
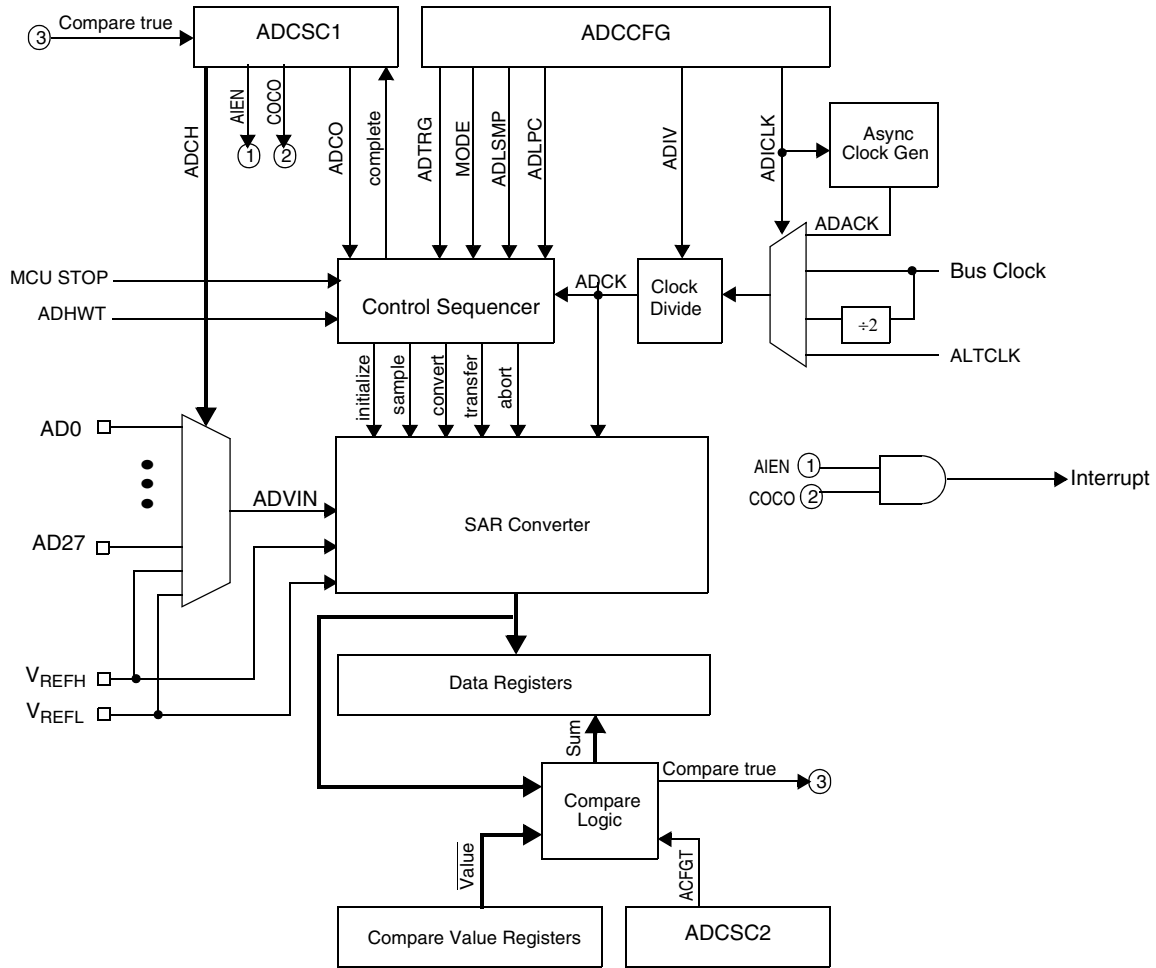
Figure 11-2. ADC Block Diagram

11.2   External Signal Description

The ADC module supports up to 28 separate analog inputs. It also requires four supply/reference/ground connections.

Table 11-2. Signal Properties

Name	Function
AD27–AD0	Analog Channel inputs
V_{REFH}	High reference voltage
V_{REFL}	Low reference voltage
V_{DDAD}	Analog power supply
V_{SSAD}	Analog ground

11.2.1 Analog Power (V$_{DDAD}$)

The ADC analog portion uses V$_{DDAD}$ as its power connection. In some packages, V$_{DDAD}$ is connected internally to V$_{DD}$. If externally available, connect the V$_{DDAD}$ pin to the same voltage potential as V$_{DD}$. External filtering may be necessary to ensure clean V$_{DDAD}$ for good results.

11.2.2 Analog Ground (V$_{SSAD}$)

The ADC analog portion uses V$_{SSAD}$ as its ground connection. In some packages, V$_{SSAD}$ is connected internally to V$_{SS}$. If externally available, connect the V$_{SSAD}$ pin to the same voltage potential as V$_{SS}$.

11.2.3 Voltage Reference High (V$_{REFH}$)

V$_{REFH}$ is the high reference voltage for the converter. In some packages, V$_{REFH}$ is connected internally to V$_{DDAD}$. If externally available, V$_{REFH}$ may be connected to the same potential as V$_{DDAD}$ or may be driven by an external source between the minimum V$_{DDAD}$ spec and the V$_{DDAD}$ potential (V$_{REFH}$ must never exceed V$_{DDAD}$).

11.2.4 Voltage Reference Low (V$_{REFL}$)

V$_{REFL}$ is the low-reference voltage for the converter. In some packages, V$_{REFL}$ is connected internally to V$_{SSAD}$. If externally available, connect the V$_{REFL}$ pin to the same voltage potential as V$_{SSAD}$.

11.2.5 Analog Channel Inputs (ADx)

The ADC module supports up to 28 separate analog inputs. An input is selected for conversion through the ADCH channel select bits.

11.3 Register Definition

These memory-mapped registers control and monitor operation of the ADC:

- Status and control register, ADCSC1
- Status and control register, ADCSC2
- Data result registers, ADCRH and ADCRL
- Compare value registers, ADCCVH and ADCCVL
- Configuration register, ADCCFG
- Pin enable registers, APCTL1, APCTL2, APCTL3

11.3.1 Status and Control Register 1 (ADCSC1)

This section describes the function of the ADC status and control register (ADCSC1). Writing ADCSC1 aborts the current conversion and initiates a new conversion (if the ADCH bits are equal to a value other than all 1s).

	7	6	5	4	3	2	1	0
R	COCO	AIEN	ADCO			ADCH		
W		AIEN	ADCO			ADCH		
Reset:	0	0	0	1	1	1	1	1

Figure 11-3. Status and Control Register (ADCSC1)

Table 11-3. ADCSC1 Field Descriptions

Field	Description
7 COCO	Conversion Complete Flag. The COCO flag is a read-only bit set each time a conversion is completed when the compare function is disabled (ACFE = 0). When the compare function is enabled (ACFE = 1), the COCO flag is set upon completion of a conversion only if the compare result is true. This bit is cleared when ADCSC1 is written or when ADCRL is read. 0 Conversion not completed 1 Conversion completed
6 AIEN	Interrupt Enable AIEN enables conversion complete interrupts. When COCO becomes set while AIEN is high, an interrupt is asserted. 0 Conversion complete interrupt disabled 1 Conversion complete interrupt enabled
5 ADCO	Continuous Conversion Enable. ADCO enables continuous conversions. 0 One conversion following a write to the ADCSC1 when software triggered operation is selected, or one conversion following assertion of ADHWT when hardware triggered operation is selected. 1 Continuous conversions initiated following a write to ADCSC1 when software triggered operation is selected. Continuous conversions are initiated by an ADHWT event when hardware triggered operation is selected.
4:0 ADCH	Input Channel Select. The ADCH bits form a 5-bit field that selects one of the input channels. The input channels are detailed in Table 11-4. The successive approximation converter subsystem is turned off when the channel select bits are all set. This feature allows for explicit disabling of the ADC and isolation of the input channel from all sources. Terminating continuous conversions this way prevents an additional, single conversion from being performed. It is not necessary to set the channel select bits to all ones to place the ADC in a low-power state when continuous conversions are not enabled because the module automatically enters a low-power state when a conversion completes.

Table 11-4. Input Channel Select

ADCH	Input Select
00000–01111	AD0–15
10000–11011	AD16–27
11100	Reserved
11101	V_{REFH}
11110	V_{REFL}
11111	Module disabled

11.3.2   Status and Control Register 2 (ADCSC2)

The ADCSC2 register controls the compare function, conversion trigger, and conversion active of the ADC module.
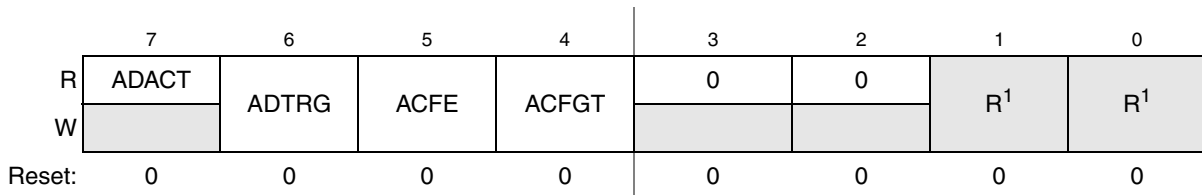
	7	6	5	4	3	2	1	0
R	ADACT	ADTRG	ACFE	ACFGT	0	0	R[1]	R[1]
W								
Reset:	0	0	0	0	0	0	0	0

Figure 11-4. Status and Control Register 2 (ADCSC2)

[1] Bits 1 and 0 are reserved bits that must always be written to 0.

Table 11-5. ADCSC2 Register Field Descriptions

Field	Description
7 ADACT	Conversion Active. Indicates that a conversion is in progress. ADACT is set when a conversion is initiated and cleared when a conversion is completed or aborted. 0 Conversion not in progress 1 Conversion in progress
6 ADTRG	Conversion Trigger Select. Selects the type of trigger used for initiating a conversion. Two types of trigger are selectable: software trigger and hardware trigger. When software trigger is selected, a conversion is initiated following a write to ADCSC1. When hardware trigger is selected, a conversion is initiated following the assertion of the ADHWT input. 0 Software trigger selected 1 Hardware trigger selected
5 ACFE	Compare Function Enable. Enables the compare function. 0 Compare function disabled 1 Compare function enabled
4 ACFGT	Compare Function Greater Than Enable. Configures the compare function to trigger when the result of the conversion of the input being monitored is greater than or equal to the compare value. The compare function defaults to triggering when the result of the compare of the input being monitored is less than the compare value. 0 Compare triggers when input is less than compare level 1 Compare triggers when input is greater than or equal to compare level

11.3.3   Data Result High Register (ADCRH)

In 12-bit operation, ADCRH contains the upper four bits of the result of a 12-bit conversion. In 10-bit mode, ADCRH contains the upper two bits of the result of a 10-bit conversion. When configured for 10-bit mode, ADR[11:10] are cleared. When configured for 8-bit mode, ADR11 – ADR8 are equal to zero.

In 12-bit and 10-bit mode, ADCRH is updated each time a conversion completes except when automatic compare is enabled and the compare condition is not met. In 12-bit and 10-bit mode, reading ADCRH prevents the ADC from transferring subsequent conversion results into the result registers until ADCRL is read. If ADCRL is not read until after the next conversion is completed, the intermediate conversion result is lost. In 8-bit mode, there is no interlocking with ADCRL.

If the MODE bits are changed, any data in ADCRH becomes invalid.

	7	6	5	4	3	2	1	0
R	0	0	0	0	ADR11	ADR10	ADR9	ADR8
W								
Reset:	0	0	0	0	0	0	0	0

Figure 11-5. Data Result High Register (ADCRH)

11.3.4   Data Result Low Register (ADCRL)

ADCRL contains the lower eight bits of the result of a 12-bit or 10-bit conversion, and all eight bits of an 8-bit conversion. This register is updated each time a conversion completes except when automatic compare is enabled and the compare condition is not met. In 12-bit and 10-bit mode, reading ADCRH prevents the ADC from transferring subsequent conversion results into the result registers until ADCRL is read. If ADCRL is not read until the after next conversion is completed, the intermediate conversion results are lost. In 8-bit mode, there is no interlocking with ADCRH. If the MODE bits are changed, any data in ADCRL becomes invalid.

	7	6	5	4	3	2	1	0
R	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 11-6. Data Result Low Register (ADCRL)

11.3.5   Compare Value High Register (ADCCVH)

In 12-bit mode, the ADCCVH register holds the upper four bits of the 12-bit compare value. These bits are compared to the upper four bits of the result following a conversion in 12-bit mode when the compare function is enabled.

	7	6	5	4	3	2	1	0
R	0	0	0	0	ADCV11	ADCV10	ADCV9	ADCV8
W								
Reset:	0	0	0	0	0	0	0	0

Figure 11-7. Compare Value High Register (ADCCVH)

In 10-bit mode, the ADCCVH register holds the upper two bits of the 10-bit compare value (ADCV9 – ADCV8). These bits are compared to the upper two bits of the result following a conversion in 10-bit mode when the compare function is enabled.

In 8-bit mode, ADCCVH is not used during compare.

11.3.6   Compare Value Low Register (ADCCVL)

This register holds the lower 8 bits of the 12-bit or 10-bit compare value or all 8 bits of the 8-bit compare value. Bits ADCV7:ADCV0 are compared to the lower 8 bits of the result following a conversion in 12-bit, 10-bit or 8-bit mode.

	7	6	5	4	3	2	1	0
R	ADCV7	ADCV6	ADCV5	ADCV4	ADCV3	ADCV2	ADCV1	ADCV0
W								
Reset:	0	0	0	0	0	0	0	0
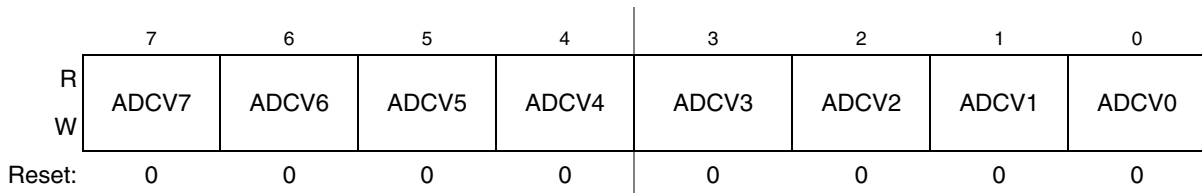
Figure 11-8. Compare Value Low Register(ADCCVL)

11.3.7   Configuration Register (ADCCFG)

ADCCFG selects the mode of operation, clock source, clock divide, and configure for low power or long sample time.

	7	6	5	4	3	2	1	0
R	ADLPC	ADIV		ADLSMP	MODE		ADICLK	
W								
Reset:	0	0	0	0	0	0	0	0
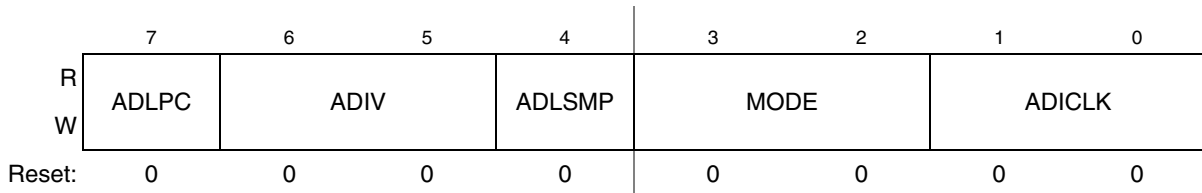
Figure 11-9. Configuration Register (ADCCFG)

Table 11-6. ADCCFG Register Field Descriptions

Field	Description
7 ADLPC	Low-Power Configuration. ADLPC controls the speed and power configuration of the successive approximation converter. This optimizes power consumption when higher sample rates are not required. 0 High speed configuration 1 Low power configuration:The power is reduced at the expense of maximum clock speed.
6:5 ADIV	Clock Divide Select. ADIV selects the divide ratio used by the ADC to generate the internal clock ADCK. Table 11-7 shows the available clock configurations.
4 ADLSMP	Long Sample Time Configuration. ADLSMP selects between long and short sample time. This adjusts the sample period to allow higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. Longer sample times can also be used to lower overall power consumption when continuous conversions are enabled if high conversion rates are not required. 0 Short sample time 1 Long sample time

Table 11-6. ADCCFG Register Field Descriptions (continued)

Field	Description
3:2 MODE	Conversion Mode Selection. MODE bits are used to select between 12-, 10-, or 8-bit operation. See Table 11-8.
1:0 ADICLK	Input Clock Select. ADICLK bits select the input clock source to generate the internal clock ADCK. See Table 11-9.

Table 11-7. Clock Divide Select

ADIV	Divide Ratio	Clock Rate
00	1	Input clock
01	2	Input clock ÷ 2
10	4	Input clock ÷ 4
11	8	Input clock ÷ 8

Table 11-8. Conversion Modes

MODE	Mode Description
00	8-bit conversion (N=8)
01	12-bit conversion (N=12)
10	10-bit conversion (N=10)
11	Reserved

Table 11-9. Input Clock Select

ADICLK	Selected Clock Source
00	Bus clock
01	Bus clock divided by 2
10	Alternate clock (ALTCLK)
11	Asynchronous clock (ADACK)

11.3.8　Pin Control 1 Register (APCTL1)

The pin control registers disable the I/O port control of MCU pins used as analog inputs. APCTL1 is

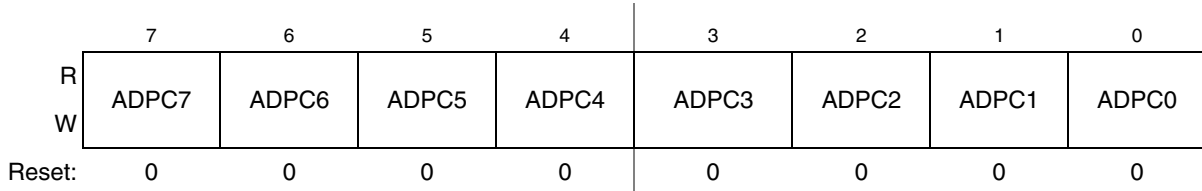used to control the pins associated with channels 0–7 of the ADC module.

	7	6	5	4	3	2	1	0
R	ADPC7	ADPC6	ADPC5	ADPC4	ADPC3	ADPC2	ADPC1	ADPC0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 11-10. Pin Control 1 Register (APCTL1)

Table 11-10. APCTL1 Register Field Descriptions

Field	Description
7 ADPC7	ADC Pin Control 7. ADPC7 controls the pin associated with channel AD7. 0 AD7 pin I/O control enabled 1 AD7 pin I/O control disabled
6 ADPC6	ADC Pin Control 6. ADPC6 controls the pin associated with channel AD6. 0 AD6 pin I/O control enabled 1 AD6 pin I/O control disabled
5 ADPC5	ADC Pin Control 5. ADPC5 controls the pin associated with channel AD5. 0 AD5 pin I/O control enabled 1 AD5 pin I/O control disabled
4 ADPC4	ADC Pin Control 4. ADPC4 controls the pin associated with channel AD4. 0 AD4 pin I/O control enabled 1 AD4 pin I/O control disabled
3 ADPC3	ADC Pin Control 3. ADPC3 controls the pin associated with channel AD3. 0 AD3 pin I/O control enabled 1 AD3 pin I/O control disabled
2 ADPC2	ADC Pin Control 2. ADPC2 controls the pin associated with channel AD2. 0 AD2 pin I/O control enabled 1 AD2 pin I/O control disabled
1 ADPC1	ADC Pin Control 1. ADPC1 controls the pin associated with channel AD1. 0 AD1 pin I/O control enabled 1 AD1 pin I/O control disabled
0 ADPC0	ADC Pin Control 0. ADPC0 controls the pin associated with channel AD0. 0 AD0 pin I/O control enabled 1 AD0 pin I/O control disabled

11.3.9 Pin Control 2 Register (APCTL2)

APCTL2 controls channels 8–15 of the ADC module.

	7	6	5	4	3	2	1	0
R	ADPC15	ADPC14	ADPC13	ADPC12	ADPC11	ADPC10	ADPC9	ADPC8
W								
Reset:	0	0	0	0	0	0	0	0

Figure 11-11. Pin Control 2 Register (APCTL2)

Table 11-11. APCTL2 Register Field Descriptions

Field	Description
7 ADPC15	ADC Pin Control 15. ADPC15 controls the pin associated with channel AD15. 0 AD15 pin I/O control enabled 1 AD15 pin I/O control disabled
6 ADPC14	ADC Pin Control 14. ADPC14 controls the pin associated with channel AD14. 0 AD14 pin I/O control enabled 1 AD14 pin I/O control disabled
5 ADPC13	ADC Pin Control 13. ADPC13 controls the pin associated with channel AD13. 0 AD13 pin I/O control enabled 1 AD13 pin I/O control disabled
4 ADPC12	ADC Pin Control 12. ADPC12 controls the pin associated with channel AD12. 0 AD12 pin I/O control enabled 1 AD12 pin I/O control disabled
3 ADPC11	ADC Pin Control 11. ADPC11 controls the pin associated with channel AD11. 0 AD11 pin I/O control enabled 1 AD11 pin I/O control disabled
2 ADPC10	ADC Pin Control 10. ADPC10 controls the pin associated with channel AD10. 0 AD10 pin I/O control enabled 1 AD10 pin I/O control disabled
1 ADPC9	ADC Pin Control 9. ADPC9 controls the pin associated with channel AD9. 0 AD9 pin I/O control enabled 1 AD9 pin I/O control disabled
0 ADPC8	ADC Pin Control 8. ADPC8 controls the pin associated with channel AD8. 0 AD8 pin I/O control enabled 1 AD8 pin I/O control disabled

11.3.10 Pin Control 3 Register (APCTL3)

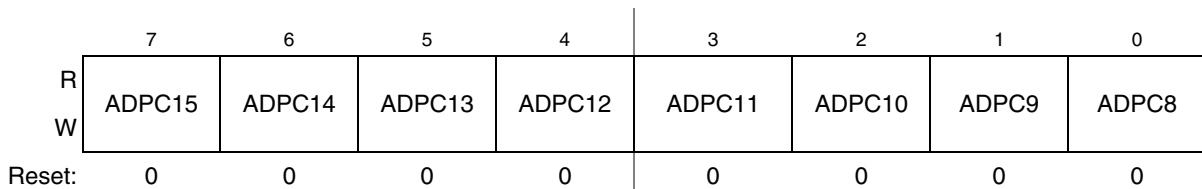APCTL3 controls channels 16–23 of the ADC module.

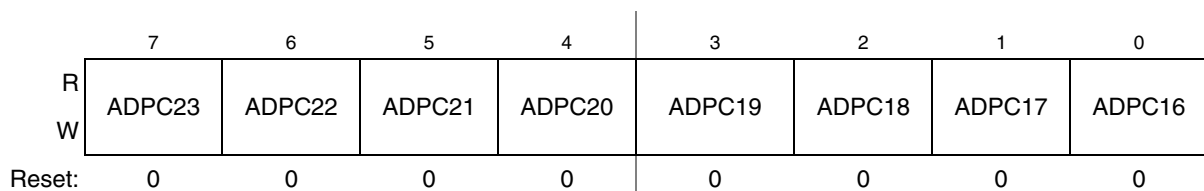|	7	6	5	4	3	2	1	0
R W	ADPC23	ADPC22	ADPC21	ADPC20	ADPC19	ADPC18	ADPC17	ADPC16
Reset:	0	0	0	0	0	0	0	0

Figure 11-12. Pin Control 3 Register (APCTL3)

Table 11-12. APCTL3 Register Field Descriptions

Field	Description
7 ADPC23	ADC Pin Control 23. ADPC23 controls the pin associated with channel AD23. 0 AD23 pin I/O control enabled 1 AD23 pin I/O control disabled
6 ADPC22	ADC Pin Control 22. ADPC22 controls the pin associated with channel AD22. 0 AD22 pin I/O control enabled 1 AD22 pin I/O control disabled
5 ADPC21	ADC Pin Control 21. ADPC21 controls the pin associated with channel AD21. 0 AD21 pin I/O control enabled 1 AD21 pin I/O control disabled
4 ADPC20	ADC Pin Control 20. ADPC20 controls the pin associated with channel AD20. 0 AD20 pin I/O control enabled 1 AD20 pin I/O control disabled
3 ADPC19	ADC Pin Control 19. ADPC19 controls the pin associated with channel AD19. 0 AD19 pin I/O control enabled 1 AD19 pin I/O control disabled
2 ADPC18	ADC Pin Control 18. ADPC18 controls the pin associated with channel AD18. 0 AD18 pin I/O control enabled 1 AD18 pin I/O control disabled
1 ADPC17	ADC Pin Control 17. ADPC17 controls the pin associated with channel AD17. 0 AD17 pin I/O control enabled 1 AD17 pin I/O control disabled
0 ADPC16	ADC Pin Control 16. ADPC16 controls the pin associated with channel AD16. 0 AD16 pin I/O control enabled 1 AD16 pin I/O control disabled

11.4 Functional Description

The ADC module is disabled during reset or when the ADCH bits are all high. The module is idle when a conversion has completed and another conversion has not been initiated. When idle, the module is in its lowest power state.

The ADC can perform an analog-to-digital conversion on any of the software selectable channels. In 12-bit and 10-bit mode, the selected channel voltage is converted by a successive approximation algorithm into a 12-bit digital result. In 8-bit mode, the selected channel voltage is converted by a successive approximation algorithm into a 9-bit digital result.

When the conversion is completed, the result is placed in the data registers (ADCRH and ADCRL). In 10-bit mode, the result is rounded to 10 bits and placed in the data registers (ADCRH and ADCRL). In 8-bit mode, the result is rounded to 8 bits and placed in ADCRL. The conversion complete flag (COCO) is then set and an interrupt is generated if the conversion complete interrupt has been enabled (AIEN = 1).

The ADC module has the capability of automatically comparing the result of a conversion with the contents of its compare registers. The compare function is enabled by setting the ACFE bit and operates with any of the conversion modes and configurations.

11.4.1   Clock Select and Divide Control

One of four clock sources can be selected as the clock source for the ADC module. This clock source is then divided by a configurable value to generate the input clock to the converter (ADCK). The clock is selected from one of the following sources by means of the ADICLK bits.

- The bus clock divided by two. For higher bus clock rates, this allows a maximum divide by 16 of the bus clock.
- ALTCLK, as defined for this MCU (See module section introduction).
- The asynchronous clock (ADACK). This clock is generated from a clock source within the ADC module. When selected as the clock source, this clock remains active while the MCU is in wait or stop3 mode and allows conversions in these modes for lower noise operation.

Whichever clock is selected, its frequency must fall within the specified frequency range for ADCK. If the available clocks are too slow, the ADC do not perform according to specifications. If the available clocks are too fast, the clock must be divided to the appropriate frequency. This divider is specified by the ADIV bits and can be divide-by 1, 2, 4, or 8.

11.4.2   Input Select and Pin Control

The pin control registers (APCTL3, APCTL2, and APCTL1) disable the I/O port control of the pins used as analog inputs.When a pin control register bit is set, the following conditions are forced for the associated MCU pin:

- The output buffer is forced to its high impedance state.
- The input buffer is disabled. A read of the I/O port returns a zero for any pin with its input buffer disabled.
- The pullup is disabled.

11.4.3   Hardware Trigger

The ADC module has a selectable asynchronous hardware conversion trigger, ADHWT, that is enabled when the ADTRG bit is set. This source is not available on all MCUs. Consult the module introduction for information on the ADHWT source specific to this MCU.

When ADHWT source is available and hardware trigger is enabled (ADTRG=1), a conversion is initiated on the rising edge of ADHWT. If a conversion is in progress when a rising edge occurs, the rising edge is ignored. In continuous convert configuration, only the initial rising edge to launch continuous conversions is observed. The hardware trigger function operates in conjunction with any of the conversion modes and configurations.

11.4.4   Conversion Control

Conversions can be performed in 12-bit mode, 10-bit mode, or 8-bit mode as determined by the MODE bits. Conversions can be initiated by a software or hardware trigger. In addition, the ADC module can be configured for low power operation, long sample time, continuous conversion, and automatic compare of the conversion result to a software determined compare value.

11.4.4.1   Initiating Conversions

A conversion is initiated:

- Following a write to ADCSC1 (with ADCH bits not all 1s) if software triggered operation is selected.
- Following a hardware trigger (ADHWT) event if hardware triggered operation is selected.
- Following the transfer of the result to the data registers when continuous conversion is enabled.

If continuous conversions are enabled, a new conversion is automatically initiated after the completion of the current conversion. In software triggered operation, continuous conversions begin after ADCSC1 is written and continue until aborted. In hardware triggered operation, continuous conversions begin after a hardware trigger event and continue until aborted.

11.4.4.2   Completing Conversions

A conversion is completed when the result of the conversion is transferred into the data result registers, ADCRH and ADCRL. This is indicated by the setting of COCO. An interrupt is generated if AIEN is high at the time that COCO is set.

A blocking mechanism prevents a new result from overwriting previous data in ADCRH and ADCRL if the previous data is in the process of being read while in 12-bit or 10-bit MODE (the ADCRH register has been read but the ADCRL register has not). When blocking is active, the data transfer is blocked, COCO is not set, and the new result is lost. In the case of single conversions with the compare function enabled and the compare condition false, blocking has no effect and ADC operation is terminated. In all other cases of operation, when a data transfer is blocked, another conversion is initiated regardless of the state of ADCO (single or continuous conversions enabled).

If single conversions are enabled, the blocking mechanism could result in several discarded conversions and excess power consumption. To avoid this issue, the data registers must not be read after initiating a single conversion until the conversion completes.

11.4.4.3   Aborting Conversions

Any conversion in progress is aborted when:

- A write to ADCSC1 occurs (the current conversion will be aborted and a new conversion will be initiated, if ADCH are not all 1s).
- A write to ADCSC2, ADCCFG, ADCCVH, or ADCCVL occurs. This indicates a mode of operation change has occurred and the current conversion is therefore invalid.
- The MCU is reset.
- The MCU enters stop mode with ADACK not enabled.

When a conversion is aborted, the contents of the data registers, ADCRH and ADCRL, are not altered. However, they continue to be the values transferred after the completion of the last successful conversion. If the conversion was aborted by a reset, ADCRH and ADCRL return to their reset states.

11.4.4.4   Power Control

The ADC module remains in its idle state until a conversion is initiated. If ADACK is selected as the conversion clock source, the ADACK clock generator is also enabled.

Power consumption when active can be reduced by setting ADLPC. This results in a lower maximum value for f_{ADCK} (see the electrical specifications).

11.4.4.5   Sample Time and Total Conversion Time

The total conversion time depends on the sample time (as determined by ADLSMP), the MCU bus frequency, the conversion mode (8-bit, 10-bit or 12-bit), and the frequency of the conversion clock (f_{ADCK}). After the module becomes active, sampling of the input begins. ADLSMP selects between short (3.5 ADCK cycles) and long (23.5 ADCK cycles) sample times.When sampling is complete, the converter is isolated from the input channel and a successive approximation algorithm is performed to determine the digital value of the analog signal. The result of the conversion is transferred to ADCRH and ADCRL upon completion of the conversion algorithm.

If the bus frequency is less than the f_{ADCK} frequency, precise sample time for continuous conversions cannot be guaranteed when short sample is enabled (ADLSMP=0). If the bus frequency is less than 1/11th of the f_{ADCK} frequency, precise sample time for continuous conversions cannot be guaranteed when long sample is enabled (ADLSMP=1).

The maximum total conversion time for different conditions is summarized in .

Table 11-13. Total Conversion Time vs. Control Conditions

Conversion Type	ADICLK	ADLSMP	Max Total Conversion Time
Single or first continuous 8-bit	0x, 10	0	20 ADCK cycles + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	0x, 10	0	23 ADCK cycles + 5 bus clock cycles
Single or first continuous 8-bit	0x, 10	1	40 ADCK cycles + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	0x, 10	1	43 ADCK cycles + 5 bus clock cycles
Single or first continuous 8-bit	11	0	5 μs + 20 ADCK + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	11	0	5 μs + 23 ADCK + 5 bus clock cycles
Single or first continuous 8-bit	11	1	5 μs + 40 ADCK + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	11	1	5 μs + 43 ADCK + 5 bus clock cycles
Subsequent continuous 8-bit; $f_{BUS} \geq f_{ADCK}$	xx	0	17 ADCK cycles
Subsequent continuous 10-bit or 12-bit; $f_{BUS} \geq f_{ADCK}$	xx	0	20 ADCK cycles
Subsequent continuous 8-bit; $f_{BUS} \geq f_{ADCK}/11$	xx	1	37 ADCK cycles
Subsequent continuous 10-bit or 12-bit; $f_{BUS} \geq f_{ADCK}/11$	xx	1	40 ADCK cycles

The maximum total conversion time is determined by the clock source chosen and the divide ratio selected. The clock source is selectable by the ADICLK bits, and the divide ratio is specified by the ADIV bits. For

example, in 10-bit mode, with the bus clock selected as the input clock source, the input clock divide-by-1 ratio selected, and a bus frequency of 8 MHz, then the conversion time for a single conversion is:

$$\text{Conversion time} = \frac{23 \text{ ADCK Cyc}}{8 \text{ MHz}/1} + \frac{5 \text{ bus Cyc}}{8 \text{ MHz}} = 3.5 \text{ ms}$$

$$\text{Number of bus cycles} = 3.5 \text{ ms} \times 8 \text{ MHz} = 28 \text{ cycles}$$

NOTE

The ADCK frequency must be between f_{ADCK} minimum and f_{ADCK} maximum to meet ADC specifications.

11.4.5 Automatic Compare Function

The compare function can be configured to check for an upper or lower limit. After the input is sampled and converted, the result is added to the two's complement of the compare value (ADCCVH and ADCCVL). When comparing to an upper limit (ACFGT = 1), if the result is greater-than or equal-to the compare value, COCO is set. When comparing to a lower limit (ACFGT = 0), if the result is less than the compare value, COCO is set. The value generated by the addition of the conversion result and the two's complement of the compare value is transferred to ADCRH and ADCRL.

Upon completion of a conversion while the compare function is enabled, if the compare condition is not true, COCO is not set and no data is transferred to the result registers. An ADC interrupt is generated upon the setting of COCO if the ADC interrupt is enabled (AIEN = 1).

NOTE

The compare function can monitor the voltage on a channel while the MCU is in wait or stop3 mode. The ADC interrupt wakes the MCU when the compare condition is met.

11.4.6 MCU Wait Mode Operation

Wait mode is a lower power-consumption standby mode from which recovery is fast because the clock sources remain active. If a conversion is in progress when the MCU enters wait mode, it continues until completion. Conversions can be initiated while the MCU is in wait mode by means of the hardware trigger or if continuous conversions are enabled.

The bus clock, bus clock divided by two, and ADACK are available as conversion clock sources while in wait mode. The use of ALTCLK as the conversion clock source in wait is dependent on the definition of ALTCLK for this MCU. Consult the module introduction for information on ALTCLK specific to this MCU.

A conversion complete event sets the COCO and generates an ADC interrupt to wake the MCU from wait mode if the ADC interrupt is enabled (AIEN = 1).

11.4.7   MCU Stop3 Mode Operation

Stop mode is a low power-consumption standby mode during which most or all clock sources on the MCU are disabled.

11.4.7.1   Stop3 Mode With ADACK Disabled

If the asynchronous clock, ADACK, is not selected as the conversion clock, executing a stop instruction aborts the current conversion and places the ADC in its idle state. The contents of ADCRH and ADCRL are unaffected by stop3 mode. After exiting from stop3 mode, a software or hardware trigger is required to resume conversions.

11.4.7.2   Stop3 Mode With ADACK Enabled

If ADACK is selected as the conversion clock, the ADC continues operation during stop3 mode. For guaranteed ADC operation, the MCU's voltage regulator must remain active during stop3 mode. Consult the module introduction for configuration information for this MCU.

If a conversion is in progress when the MCU enters stop3 mode, it continues until completion. Conversions can be initiated while the MCU is in stop3 mode by means of the hardware trigger or if continuous conversions are enabled.

A conversion complete event sets the COCO and generates an ADC interrupt to wake the MCU from stop3 mode if the ADC interrupt is enabled (AIEN = 1).

NOTE

> The ADC module can wake the system from low-power stop and cause the MCU to begin consuming run-level currents without generating a system level interrupt. To prevent this scenario, software should ensure the data transfer blocking mechanism (discussed in Section 11.4.4.2, "Completing Conversions) is cleared when entering stop3 and continuing ADC conversions.

11.4.8   MCU Stop2 Mode Operation

The ADC module is automatically disabled when the MCU enters stop2 mode. All module registers contain their reset values following exit from stop2. Therefore, the module must be re-enabled and re-configured following exit from stop2.

11.5   Initialization Information

This section gives an example that provides some basic direction on how to initialize and configure the ADC module. You can configure the module for 8-, 10-, or 12-bit resolution, single or continuous conversion, and a polled or interrupt approach, among many other options. Refer to Table 11-7, Table 11-8, and Table 11-9 for information used in this example.

NOTE

Hexadecimal values designated by a preceding 0x, binary values designated
by a preceding %, and decimal values have no preceding character.

11.5.1   ADC Module Initialization Example

11.5.1.1   Initialization Sequence

Before the ADC module can be used to complete conversions, an initialization procedure must be
performed. A typical sequence is as follows:

1. Update the configuration register (ADCCFG) to select the input clock source and the divide ratio
   used to generate the internal clock, ADCK. This register is also used for selecting sample time and
   low-power configuration.

2. Update status and control register 2 (ADCSC2) to select the conversion trigger (hardware or
   software) and compare function options, if enabled.

3. Update status and control register 1 (ADCSC1) to select whether conversions will be continuous
   or completed only once, and to enable or disable conversion complete interrupts. The input channel
   on which conversions will be performed is also selected here.

11.5.1.2   Pseudo-Code Example

In this example, the ADC module is set up with interrupts enabled to perform a single 10-bit conversion
at low power with a long sample time on input channel 1, where the internal ADCK clock is derived from
the bus clock divided by 1.

ADCCFG = 0x98 (%10011000)

```
        Bit 7    ADLPC    1         Configures for low power (lowers maximum clock speed)
        Bit 6:5  ADIV     00        Sets the ADCK to the input clock ÷ 1
        Bit 4    ADLSMP   1         Configures for long sample time
        Bit 3:2  MODE     10        Sets mode at 10-bit conversions
        Bit 1:0  ADICLK   00        Selects bus clock as input clock source
```

ADCSC2 = 0x00 (%00000000)

```
        Bit 7    ADACT    0         Flag indicates if a conversion is in progress
        Bit 6    ADTRG    0         Software trigger selected
        Bit 5    ACFE     0         Compare function disabled
        Bit 4    ACFGT    0         Not used in this example
        Bit 3:2           00        Reserved, always reads zero
        Bit 1:0           00        Reserved for Freescale's internal use; always write zero
```

ADCSC1 = 0x41 (%01000001)

```
        Bit 7    COCO     0         Read-only flag which is set when a conversion completes
        Bit 6    AIEN     1         Conversion complete interrupt enabled
        Bit 5    ADCO     0         One conversion only (continuous conversions disabled)
        Bit 4:0  ADCH     00001     Input channel 1 selected as ADC input channel
```

ADCRH/L = 0xxx

```
        Holds results of conversion. Read high byte (ADCRH) before low byte (ADCRL) so that
        conversion data cannot be overwritten with data from the next conversion.
```

MCF51QE128 MCU Series Reference Manual, Rev. 3

Get the latest version from freescale.com

ADCCVH/L = 0xxx

> Holds compare value when compare function enabled

APCTL1=0x02

> AD1 pin I/O control disabled. All other AD pins remain general purpose I/O pins

APCTL2=0x00

> All other AD pins remain general purpose I/O pins
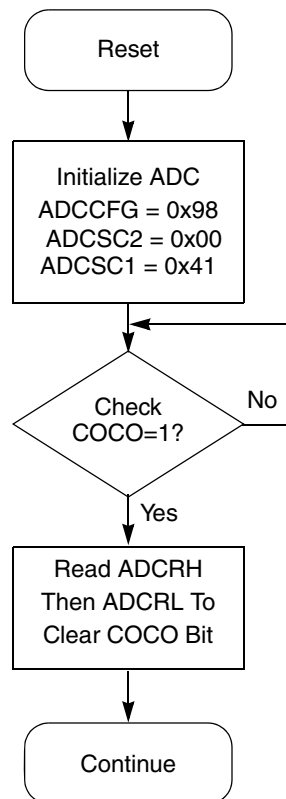
Figure 11-13. Initialization Flowchart for Example

11.6   Application Information

This section contains information for using the ADC module in applications. The ADC has been designed to be integrated into a microcontroller for use in embedded control applications requiring an A/D converter.

11.6.1   External Pins and Routing

The following sections discuss the external pins associated with the ADC module and how they should be used for best results.

11.6.1.1   Analog Supply Pins

The ADC module has analog power and ground supplies (V_{DDAD} and V_{SSAD}) available as separate pins on some devices. V_{SSAD} is shared on the same pin as the MCU digital V_{SS} on some devices. On other devices, V_{SSAD} and V_{DDAD} are shared with the MCU digital supply pins. In these cases, there are separate pads for the analog supplies bonded to the same pin as the corresponding digital supply so that some degree of isolation between the supplies is maintained.

When available on a separate pin, both V_{DDAD} and V_{SSAD} must be connected to the same voltage potential as their corresponding MCU digital supply (V_{DD} and V_{SS}) and must be routed carefully for maximum noise immunity and bypass capacitors placed as near as possible to the package.

If separate power supplies are used for analog and digital power, the ground connection between these supplies must be at the V_{SSAD} pin. This should be the only ground connection between these supplies if possible. The V_{SSAD} pin makes a good single point ground location.

11.6.1.2   Analog Reference Pins

In addition to the analog supplies, the ADC module has connections for two reference voltage inputs. The high reference is V_{REFH}, which may be shared on the same pin as V_{DDAD} on some devices. The low reference is V_{REFL}, which may be shared on the same pin as V_{SSAD} on some devices.

When available on a separate pin, V_{REFH} may be connected to the same potential as V_{DDAD}, or may be driven by an external source between the minimum V_{DDAD} spec and the V_{DDAD} potential (V_{REFH} must never exceed V_{DDAD}). When available on a separate pin, V_{REFL} must be connected to the same voltage potential as V_{SSAD}. V_{REFH} and V_{REFL} must be routed carefully for maximum noise immunity and bypass capacitors placed as near as possible to the package.

AC current in the form of current spikes required to supply charge to the capacitor array at each successive approximation step is drawn through the V_{REFH} and V_{REFL} loop. The best external component to meet this current demand is a 0.1 μF capacitor with good high frequency characteristics. This capacitor is connected between V_{REFH} and V_{REFL} and must be placed as near as possible to the package pins. Resistance in the path is not recommended because the current causes a voltage drop that could result in conversion errors. Inductance in this path must be minimum (parasitic only).

11.6.1.3   Analog Input Pins

The external analog inputs are typically shared with digital I/O pins on MCU devices. The pin I/O control is disabled by setting the appropriate control bit in one of the pin control registers. Conversions can be performed on inputs without the associated pin control register bit set. It is recommended that the pin control register bit always be set when using a pin as an analog input. This avoids problems with contention because the output buffer is in its high impedance state and the pullup is disabled. Also, the input buffer draws DC current when its input is not at V_{DD} or V_{SS}. Setting the pin control register bits for all pins used as analog inputs should be done to achieve lowest operating current.

Empirical data shows that capacitors on the analog inputs improve performance in the presence of noise or when the source impedance is high. Use of 0.01 μF capacitors with good high-frequency characteristics is sufficient. These capacitors are not necessary in all cases, but when used they must be placed as near as possible to the package pins and be referenced to V_{SSA}.

For proper conversion, the input voltage must fall between V_{REFH} and V_{REFL}. If the input is equal to or exceeds V_{REFH}, the converter circuit converts the signal to 0xFFF (full scale 12-bit representation), 0x3FF (full scale 10-bit representation) or 0xFF (full scale 8-bit representation). If the input is equal to or less than V_{REFL}, the converter circuit converts it to 0x000. Input voltages between V_{REFH} and V_{REFL} are straight-line linear conversions. There is a brief current associated with V_{REFL} when the sampling capacitor is charging. The input is sampled for 3.5 cycles of the ADCK source when ADLSMP is low, or 23.5 cycles when ADLSMP is high.

For minimal loss of accuracy due to current injection, pins adjacent to the analog input pins should not be transitioning during conversions.

11.6.2 Sources of Error

Several sources of error exist for A/D conversions. These are discussed in the following sections.

11.6.2.1 Sampling Error

For proper conversions, the input must be sampled long enough to achieve the proper accuracy. Given the maximum input resistance of approximately 7kΩ and input capacitance of approximately 5.5 pF, sampling to within 1/4LSB (at 12-bit resolution) can be achieved within the minimum sample window (3.5 cycles @ 8 MHz maximum ADCK frequency) provided the resistance of the external analog source (R_{AS}) is kept below 2 kΩ.

Higher source resistances or higher-accuracy sampling is possible by setting ADLSMP (to increase the sample window to 23.5 cycles) or decreasing ADCK frequency to increase sample time.

11.6.2.2 Pin Leakage Error

Leakage on the I/O pins can cause conversion error if the external analog source resistance (R_{AS}) is high. If this error cannot be tolerated by the application, keep R_{AS} lower than $V_{DDAD} / (2^N * I_{LEAK})$ for less than 1/4LSB leakage error (N = 8 in 8-bit, 10 in 10-bit or 12 in 12-bit mode).

11.6.2.3 Noise-Induced Errors

System noise that occurs during the sample or conversion process can affect the accuracy of the conversion. The ADC accuracy numbers are guaranteed as specified only if the following conditions are met:

- There is a 0.1 µF low-ESR capacitor from V_{REFH} to V_{REFL}.
- There is a 0.1 µF low-ESR capacitor from V_{DDAD} to V_{SSAD}.
- If inductive isolation is used from the primary supply, an additional 1 µF capacitor is placed from V_{DDAD} to V_{SSAD}.
- V_{SSAD} (and V_{REFL}, if connected) is connected to V_{SS} at a quiet point in the ground plane.
- Operate the MCU in wait or stop3 mode before initiating (hardware triggered conversions) or immediately after initiating (hardware or software triggered conversions) the ADC conversion.
  - For software triggered conversions, immediately follow the write to ADCSC1 with a stop instruction.

MCF51QE128 MCU Series Reference Manual, Rev. 3

Freescale Semiconductor

— For stop3 mode operation, select ADACK as the clock source. Operation in stop3 reduces V_{DD} noise but increases effective conversion time due to stop recovery.

- There is no I/O switching, input or output, on the MCU during the conversion.

There are some situations where external system activity causes radiated or conducted noise emissions or excessive V_{DD} noise is coupled into the ADC. In these situations, or when the MCU cannot be placed in wait or stop3 or I/O activity cannot be halted, these recommended actions may reduce the effect of noise on the accuracy:

- Place a 0.01 µF capacitor (C_{AS}) on the selected input channel to V_{REFL} or V_{SSAD} (this improves noise issues, but affects the sample rate based on the external analog source resistance).

- Average the result by converting the analog input many times in succession and dividing the sum of the results. Four samples are required to eliminate the effect of a 1LSB, one-time error.

- Reduce the effect of synchronous noise by operating off the asynchronous clock (ADACK) and averaging. Noise that is synchronous to ADCK cannot be averaged out.

11.6.2.4 Code Width and Quantization Error

The ADC quantizes the ideal straight-line transfer function into 4096 steps (in 12-bit mode). Each step ideally has the same height (1 code) and width. The width is defined as the delta between the transition points to one code and the next. The ideal code width for an N bit converter (in this case N can be 8, 10 or 12), defined as 1LSB, is:

$$1 \text{ lsb} = (V_{REFH} - V_{REFL}) / 2^N$$

Eqn. 11-2

There is an inherent quantization error due to the digitization of the result. For 8-bit or 10-bit conversions the code transitions when the voltage is at the midpoint between the points where the straight line transfer function is exactly represented by the actual transfer function. Therefore, the quantization error will be ± 1/2 lsb in 8- or 10-bit mode. As a consequence, however, the code width of the first (0x000) conversion is only 1/2 lsb and the code width of the last (0xFF or 0x3FF) is 1.5 lsb.

For 12-bit conversions the code transitions only after the full code width is present, so the quantization error is −1 lsb to 0 lsb and the code width of each step is 1 lsb.

11.6.2.5 Linearity Errors

The ADC may also exhibit non-linearity of several forms. Every effort has been made to reduce these errors but the system should be aware of them because they affect overall accuracy. These errors are:

- Zero-scale error (E_{ZS}) (sometimes called offset) — This error is defined as the difference between the actual code width of the first conversion and the ideal code width (1/2 lsb in 8-bit or 10-bit modes and 1 lsb in 12-bit mode). If the first conversion is 0x001, the difference between the actual 0x001 code width and its ideal (1 lsb) is used.

- Full-scale error (E_{FS}) — This error is defined as the difference between the actual code width of the last conversion and the ideal code width (1.5 lsb in 8-bit or 10-bit modes and 1LSB in 12-bit mode). If the last conversion is 0x3FE, the difference between the actual 0x3FE code width and its ideal (1LSB) is used.

- Differential non-linearity (DNL) — This error is defined as the worst-case difference between the actual code width and the ideal code width for all conversions.

- Integral non-linearity (INL) — This error is defined as the highest-value the (absolute value of the) running sum of DNL achieves. More simply, this is the worst-case difference of the actual transition voltage to a given code and its corresponding ideal transition voltage, for all codes.

- Total unadjusted error (TUE) — This error is defined as the difference between the actual transfer function and the ideal straight-line transfer function and includes all forms of error.

11.6.2.6   Code Jitter, Non-Monotonicity, and Missing Codes

Analog-to-digital converters are susceptible to three special forms of error. These are code jitter, non-monotonicity, and missing codes.

Code jitter is when, at certain points, a given input voltage converts to one of two values when sampled repeatedly. Ideally, when the input voltage is infinitesimally smaller than the transition voltage, the converter yields the lower code (and vice-versa). However, even small amounts of system noise can cause the converter to be indeterminate (between two codes) for a range of input voltages around the transition voltage. This range is normally around ±1/2 lsb in 8-bit or 10-bit mode, or around 2 lsb in 12-bit mode, and increases with noise.

This error may be reduced by repeatedly sampling the input and averaging the result. Additionally the techniques discussed in Section 11.6.2.3 reduces this error.

Non-monotonicity is defined as when, except for code jitter, the converter converts to a lower code for a higher input voltage. Missing codes are those values never converted for any input value.

In 8-bit or 10-bit mode, the ADC is guaranteed to be monotonic and have no missing codes.

Chapter 12
Internal Clock Source (S08ICSV3)

12.1   Introduction

The internal clock source (ICS) module provides clock source choices for the MCU. The module contains a frequency-locked loop (FLL) as a clock source that is controllable by either an internal or an external reference clock. The module can provide this FLL clock or either of the internal or external reference clocks as a source for the MCU system clock.

The ICSTRM and FTRIM bits are normally reset to the factory trim values on any reset. However, any reset that puts the device into BDM (a POR with the BKGD pin held low or a development tool setting SBDFR[BDFR]) results in the ICSTRM and FTRIM bits being set to values of 0x80 and 0. When debugging the MCU, the factory trim value can be used by copying the trim values from the Flash locations shown in table 4-4.

There are also signals provided to control a low power oscillator (XOSCVLP) module to allow the use of an external crystal/resonator as the external reference clock.

Whichever clock source is chosen, it is passed through a reduced bus divider (BDIV) which allows a lower final output clock frequency to be derived.

12.1.1   External Oscillator

The external oscillator module (XOSCVLP) provides the external clock opitons to the ICS module. The output of this submodule (OSCOUT) can be used as the real-time counter module (RTC) clock source.

12.1.2   Stop2 Mode Considerations

If you are using a low range oscillator during stop2, reconfigure the ICSC2 register (the oscillator control bits) before PPDACK is written. The low range (RANGE=0) oscillator can operate in stop2 to be the clock source for the RTC module. If the low range oscillator is active when entering stop2, it remains active in stop2 regardless of the value of EREFSTEN. To disable the oscillator in stop2, switch the ICS into FBI or FEI mode before executing the STOP instruction.

Figure 12-1 shows the MCF51QE128 Series block diagram with the ICS highlighted.
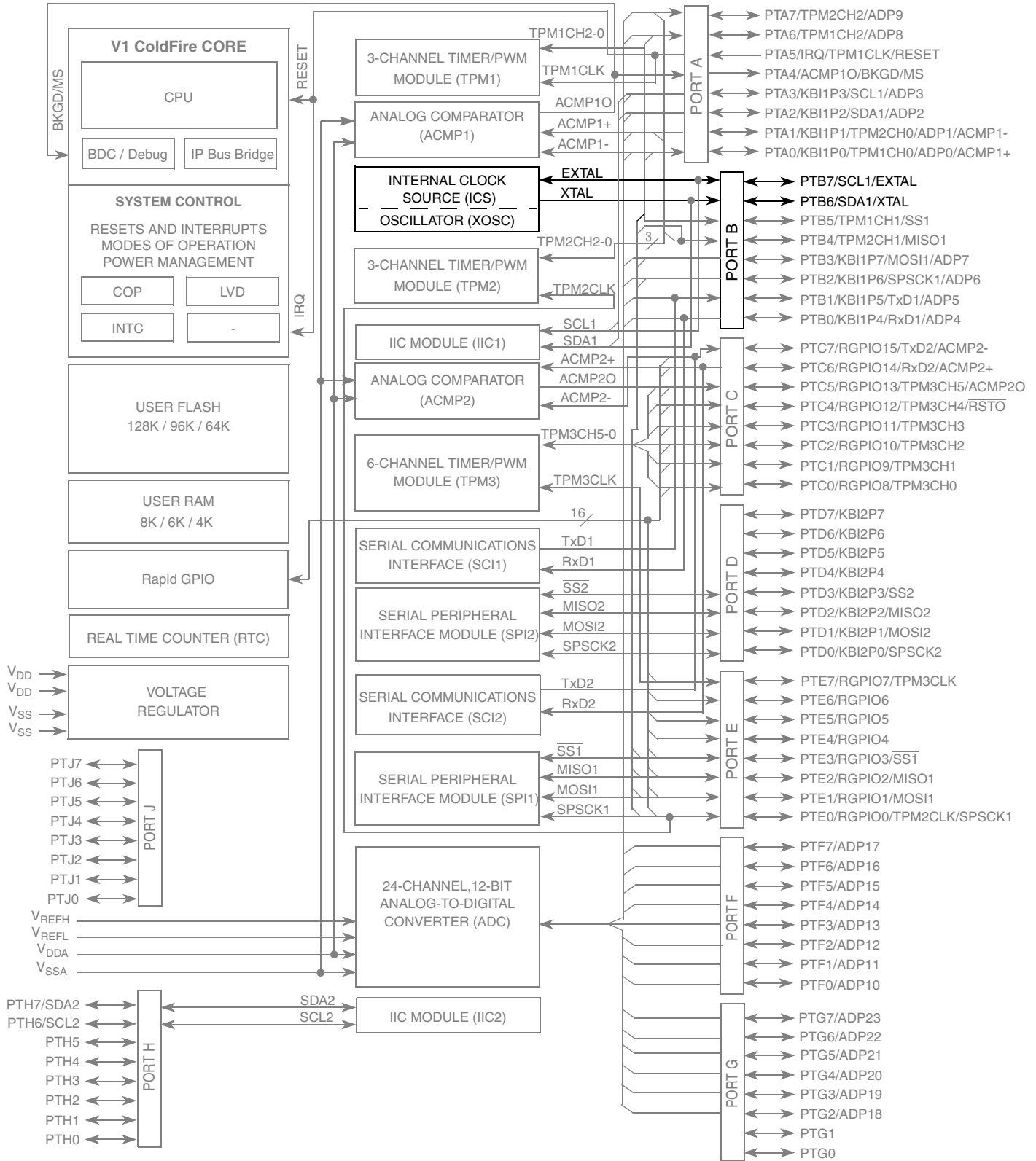
Figure 12-1. MCF51QE128 Series Block Diagram Highlighting ICS Block and Pins

12.1.3   Features

Key features of the ICS module are:

- Frequency-locked loop (FLL) is trimmable for accuracy
- Internal or external reference clocks can be used to control the FLL
- Reference divider is provided for external clock
- Internal reference clock has 9 trim bits available
- Internal or external reference clocks can be selected as the clock source for the MCU
- Whichever clock is selected as the source can be divided down
  — 2 bit select for clock divider is provided
    – Allowable dividers are: 1, 2, 4, 8
- Control signals for a low power oscillator as the external reference clock are provided
  — HGO, RANGE, EREFS, ERCLKEN, EREFSTEN
- FLL Engaged Internal mode is automatically selected out of reset
- BDC clock is provided as a constant divide by 2 of the low range DCO output
- Three selectable digitally controlled oscillators (DCO) optimized for different frequency ranges.
- Option to maximize output frequency for a 32768 Hz external reference clock source.

12.1.4 Block Diagram

Figure 12-2 is the ICS block diagram.

Figure 12-2. Internal Clock Source (ICS) Block Diagram

12.1.5 Modes of Operation

There are seven modes of operation for the ICS: FEI, FEE, FBI, FBILP, FBE, FBELP, and stop.

12.1.5.1 FLL Engaged Internal (FEI)

In FLL engaged internal mode, which is the default mode, the ICS supplies a clock derived from the FLL controlled by the internal reference clock. The BDC clock is supplied from the FLL.

12.1.5.2 FLL Engaged External (FEE)

In FLL engaged external mode, the ICS supplies a clock derived from the FLL controlled by an external reference clock. The BDC clock is supplied from the FLL.

12.1.5.3 FLL Bypassed Internal (FBI)

In FLL bypassed internal mode, the FLL is enabled and controlled by the internal reference clock, but is bypassed. The ICS supplies a clock derived from the internal reference clock. The BDC clock is supplied from the FLL.

12.1.5.4 FLL Bypassed Internal Low Power (FBILP)

In FLL bypassed internal low-power mode, the FLL is disabled and bypassed, and the ICS supplies a clock derived from the internal reference clock. The BDC clock is not available.

12.1.5.5 FLL Bypassed External (FBE)

In FLL bypassed external mode, the FLL is enabled and controlled by an external reference clock, but is bypassed. The ICS supplies a clock derived from the external reference clock. The external reference clock can be an external crystal/resonator supplied by an OSC controlled by the ICS, or it can be another external clock source. The BDC clock is supplied from the FLL.

12.1.5.6 FLL Bypassed External Low Power (FBELP)

In FLL bypassed external low-power mode, the FLL is disabled and bypassed, and the ICS supplies a clock derived from the external reference clock. The external reference clock can be an external crystal/resonator supplied by an OSC controlled by the ICS, or it can be another external clock source. The BDC clock is not available.

12.1.5.7 Stop (STOP)

In stop mode, the FLL is disabled and the internal or external reference clocks can be selected to be enabled or disabled. The BDC clock is not available and the ICS does not provide an MCU clock source.

12.2 External Signal Description

There are no ICS signals that connect off chip.

12.3 Register Definition

Figure 12-1 is a summary of ICS registers.

Table 12-1. ICS Register Summary

Name		7	6	5	4	3	2	1	0
ICSC1	R	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
	W								
ICSC2	R	BDIV		RANGE	HGO	LP	EREFS	ERCLKEN	EREFSTEN
	W								
ICSTRM	R	TRIM							
	W								
ICSSC	R	DRST		DMX32	IREFST	CLKST		OSCINIT	FTRIM
	W	DRS							

MCF51QE128 MCU Series Reference Manual, Rev. 3

12.3.1   ICS Control Register 1 (ICSC1)

	7	6	5	4	3	2	1	0
R	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
W								
Reset:	0	0	0	0	0	1	0	0
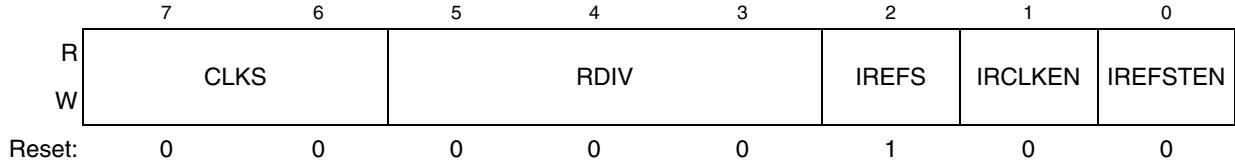
Figure 12-3. ICS Control Register 1 (ICSC1)

Table 12-2. ICSC1 Field Descriptions

Field	Description
7:6 CLKS	Clock Source Select. Selects the clock source that controls the bus frequency. The actual bus frequency depends on the value of the BDIV bits. 00 Output of FLL is selected. 01 Internal reference clock is selected. 10 External reference clock is selected. 11 Reserved, defaults to 00.
5:3 RDIV	Reference Divider. Selects the amount to divide down the external reference clock. Resulting frequency must be in the range 31.25 kHz to 39.0625 kHz. See Table 12-3 for the divide-by factors.
2 IREFS	Internal Reference Select. The IREFS bit selects the reference clock source for the FLL. 1 Internal reference clock selected 0 External reference clock selected
1 IRCLKEN	Internal Reference Clock Enable. The IRCLKEN bit enables the internal reference clock for use as ICSIRCLK. 1 ICSIRCLK active 0 ICSIRCLK inactive
0 IREFSTEN	Internal Reference Stop Enable. The IREFSTEN bit controls whether or not the internal reference clock remains enabled when the ICS enters stop mode. 1 Internal reference clock stays enabled in stop if IRCLKEN is set before entering stop 0 Internal reference clock is disabled in stop

Table 12-3. Reference Divide Factor

RDIV	RANGE=0	RANGE=1
0	1[1]	32
1	2	64
2	4	128
3	8	256
4	16	512
5	32	1024
6	64	Reserved
7	128	Reserved

[1]   Reset default

12.3.2 ICS Control Register 2 (ICSC2)

	7	6	5	4	3	2	1	0
R	BDIV		RANGE	HGO	LP	EREFS	ERCLKEN	EREFSTEN
W								
Reset:	0	1	0	0	0	0	0	0

Figure 12-4. ICS Control Register 2 (ICSC2)

Table 12-4. ICSC2 Field Descriptions

Field	Description
7:6 BDIV	Bus Frequency Divider. Selects the amount to divide down the clock source selected by the CLKS bits. This controls the bus frequency. 00 Encoding 0 — Divides selected clock by 1 01 Encoding 1 — Divides selected clock by 2 (reset default) 10 Encoding 2 — Divides selected clock by 4 11 Encoding 3 — Divides selected clock by 8
5 RANGE	Frequency Range Select. Selects the frequency range for the external oscillator. 1 High frequency range selected for the external oscillator 0 Low frequency range selected for the external oscillator
4 HGO	High Gain Oscillator Select. The HGO bit controls the external oscillator mode of operation. 1 Configure external oscillator for high gain operation 0 Configure external oscillator for low power operation
3 LP	Low Power Select. The LP bit controls whether the FLL is disabled in FLL bypassed modes. 1 FLL is disabled in bypass modes unless BDM is active 0 FLL is not disabled in bypass mode
2 EREFS	External Reference Select. The EREFS bit selects the source for the external reference clock. 1 Oscillator requested 0 External Clock Source requested
1 ERCLKEN	External Reference Enable. The ERCLKEN bit enables the external reference clock for use as ICSERCLK. 1 ICSERCLK active 0 ICSERCLK inactive
0 EREFSTEN	External Reference Stop Enable. The EREFSTEN bit controls whether or not the external reference clock remains enabled when the ICS enters stop mode. 1 External reference clock stays enabled in stop if ERCLKEN is set before entering stop 0 External reference clock is disabled in stop

12.3.3 ICS Trim Register (ICSTRM)

	7	6	5	4	3	2	1	0
R				TRIM				
W								
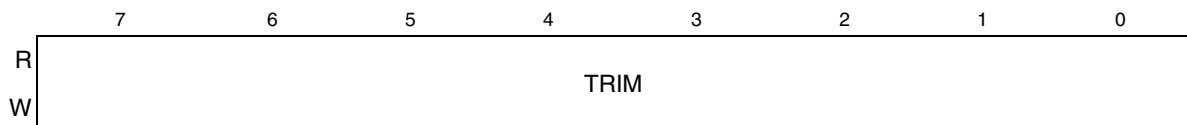
Reset: **Note:** TRIM is loaded during reset from a factory programmed location when not in BDM mode. If in a BDM mode, a default value of 0x80 is loaded.

Figure 12-5. ICS Trim Register (ICSTRM)

Table 12-5. ICSTRM Field Descriptions

Field	Description
7:0 TRIM	ICS Trim Setting. The TRIM bits control the internal reference clock frequency by controlling the internal reference clock period. The bits' affects are binary weighted (bit 1 adjusts twice as much as bit 0). Increasing the binary value in TRIM increases the period and decreasing the value decreases the period. An additional fine trim bit is available in ICSSC as the FTRIM bit.

12.3.4   ICS Status and Control (ICSSC)

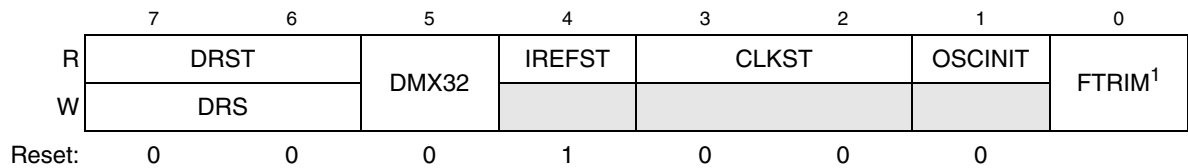|	7	6	5	4	3	2	1	0
R	DRST		DMX32	IREFST	CLKST		OSCINIT	FTRIM[1]
W	DRS							
Reset:	0	0	0	1	0	0	0	

Figure 12-6. ICS Status and Control Register (ICSSC)

[1] FTRIM is loaded during reset from a factory programmed location when not in any BDM mode. If in a BDM mode, FTRIM gets loaded with a value of 1'b0.

Table 12-6. ICSSC Field Descriptions

Field	Description
7-6 DRST DRS	DCO Range Status. The DRST read field indicates the current frequency range for the FLL output, DCOOUT. See Table 12-7. The DRST field does not update immediately after a write to the DRS field due to internal synchronization between clock domains. Writing the DRS bits to 2'b11 is ignored and the DRST bits remain with the current setting. DCO Range Select. The DRS field selects the frequency range for the FLL output, DCOOUT. Writes to the DRS field while the LP bit is set are ignored. 00 Low range. 01 Mid range. 10 High range. 11 Reserved.
5 DMX32	DCO Maximum Frequency with 32.768 kHz Reference. The DMX32 bit controls whether or not the DCO frequency range is narrowed to its maximum frequency with a 32.768 kHz reference. See Table 12-7. 0 DCO has default range of 25%. 1 DCO is fined tuned for maximum frequency with 32.768 kHz reference.
4 IREFST	Internal Reference Status. The IREFST bit indicates the current source for the reference clock. The IREFST bit does not update immediately after a write to the IREFS bit due to internal synchronization between clock domains. 0 Source of reference clock is external clock. 1 Source of reference clock is internal clock.
3-2 CLKST	Clock Mode Status. The CLKST bits indicate the current clock mode. The CLKST bits don't update immediately after a write to the CLKS bits due to internal synchronization between clock domains. 00 Output of FLL is selected. 01 FLL Bypassed, Internal reference clock is selected. 10 FLL Bypassed, External reference clock is selected. 11 Reserved.

Table 12-6. ICSSC Field Descriptions (continued)

Field	Description
1 OSCINIT	OSC Initialization. If EREFS is set and the external reference clock is selected by ERCLKEN or by the ICS being in FEE, FBE, or FBELP mode, this bit is set after the initialization cycles of the external oscillator clock have completed. This bit is only cleared when either ERCLKEN or EREFS are cleared.
0 FTRIM	ICS Fine Trim. The FTRIM bit controls the smallest adjustment of the internal reference clock frequency. Setting FTRIM increases the period and clearing FTRIM decreases the period by the smallest amount possible.

Table 12-7. DCO Frequency Range[1]

DRS	DMX32	Reference range	FLL factor	DCO range
00	0	31.25 – 39.0625 kHz	512	16 – 20 MHz
	1	32.768 kHz	608	19.92 MHz
01	0	31.25 – 39.0625 kHz	1024	32 – 40 MHz
	1	32.768 kHz	1216	39.85 MHz
10	0	31.25 – 39.0625 kHz	1536	48 – 60 MHz
	1	32.768 kHz	1824	59.77 MHz
11	Reserved			

[1] The resulting bus clock frequency should not exceed the maximum specified bus clock frequency of the device.

12.4 Functional Description
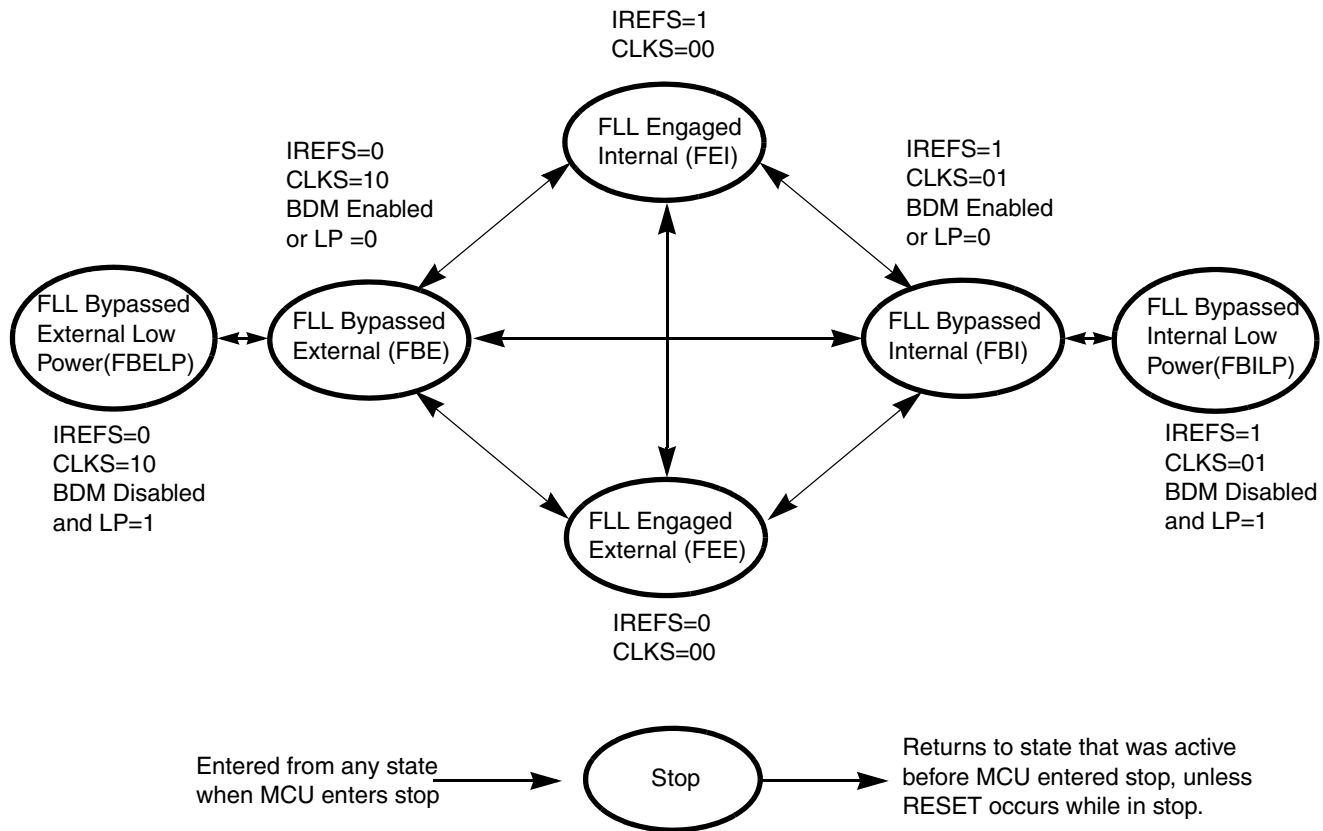
12.4.1 Operational Modes

Figure 12-7. Clock Switching Modes

The seven states of the ICS are shown as a state diagram and are described below. The arrows indicate the allowed movements between the states.

12.4.1.1 FLL Engaged Internal (FEI)

FLL engaged internal (FEI) is the default mode of operation and entered when all the following conditions occur:

- CLKS bits are written to 00.
- IREFS bit is written to 1.

In FLL engaged internal mode, the ICSOUT clock is derived from the FLL clock controlled by the internal reference clock. The FLL loop locks the frequency to the FLL factor times the internal reference frequency. The ICSLCLK is available for BDC communications, and the internal reference clock is enabled.

12.4.1.2 FLL Engaged External (FEE)

The FLL engaged external (FEE) mode is entered when all the following conditions occur:

- CLKS bits are written to 00.
- IREFS bit is written to 0.
- RDIV bits are written to divide external reference clock to be within the range of 31.25 kHz to 39.0625 kHz.

In FLL engaged external mode, the ICSOUT clock is derived from the FLL clock controlled by the external reference clock.The FLL loop locks the frequency to the FLL factor times the external reference frequency, as selected by the RDIV bits. The ICSLCLK is available for BDC communications, and the external reference clock is enabled.

12.4.1.3 FLL Bypassed Internal (FBI)

The FLL bypassed internal (FBI) mode is entered when all the following conditions occur:

- CLKS bits are written to 01.
- IREFS bit is written to 1.
- BDM mode is active or LP bit is written to 0.

In FLL bypassed internal mode, the ICSOUT clock is derived from the internal reference clock. The internal reference clock controls the FLL clock, and the FLL loop locks the FLL frequency to the FLL factor times the internal reference frequency. The ICSLCLK is available for BDC communications, and the internal reference clock is enabled.

12.4.1.4 FLL Bypassed Internal Low Power (FBILP)

The FLL bypassed internal low-power (FBILP) mode is entered when all the following conditions occur:

- CLKS bits are written to 01
- IREFS bit is written to 1.
- BDM mode is not active and LP bit is written to 1

In FLL bypassed internal low-power mode, the ICSOUT clock is derived from the internal reference clock and the FLL is disabled. The ICSLCLK is not available for BDC communications, and the internal reference clock is enabled.

12.4.1.5 FLL Bypassed External (FBE)

The FLL bypassed external (FBE) mode is entered when all the following conditions occur:

- CLKS bits are written to 10.
- IREFS bit is written to 0.
- RDIV bits are written to divide external reference clock to be within the range of 31.25 kHz to 39.0625 kHz.
- BDM mode is active or LP bit is written to 0.

In FLL bypassed external mode, the ICSOUT clock is derived from the external reference clock. The external reference clock controls the FLL clock, and the FLL loop locks the FLL frequency to the FLL factor times the external reference frequency, as selected by the RDIV bits. The ICSLCLK is available for BDC communications, and the external reference clock is enabled.

12.4.1.6   FLL Bypassed External Low Power (FBELP)

The FLL bypassed external low-power (FBELP) mode is entered when all the following conditions occur:

- CLKS bits are written to 10.
- IREFS bit is written to 0.
- BDM mode is not active and LP bit is written to 1.

In FLL bypassed external low power mode, the ICSOUT clock is derived from the external reference clock and the FLL is disabled. The ICSLCLK is not available for BDC communications. The external reference clock is enabled.

12.4.1.7   Stop

Stop mode is entered when the MCU enters a stop state. In this mode, all ICS clock signals are static except in the following cases:

ICSIRCLK is active in stop mode when all the following conditions occur:

- IRCLKEN bit is written to 1
- IREFSTEN bit is written to 1

ICSERCLK is active in stop mode when all the following conditions occur:

- ERCLKEN bit is written to 1EREFSTEN bit is written to 1

12.4.2   Mode Switching

The IREF bit can be changed at anytime, but the actual switch to the newly selected clock is shown by the IREFST bit. When switching between FLL engaged internal (FEI) and FLL engaged external (FEE) modes, the FLL begins locking again after the switch is completed.

The CLKS bits can also be changed at anytime, but the actual switch to the newly selected clock is shown by the CLKST bits. If the newly selected clock is not available, the previous clock remains selected.

The DRS bits can be changed at anytime except when LP bit is 1. If the DRS bits are changed while in FLL engaged internal (FEI) or FLL engaged external (FEE), the bus clock remains at the previous DCO range until the new DCO starts. When the new DCO starts the bus clock switches to it. After switching to the new DCO ,the FLL remains unlocked for several reference cycles. After the selected DCO startup time is over, the FLL is locked. The completion of the switch is shown by the DRST bits.

12.4.3   Bus Frequency Divider

The BDIV bits can be changed at anytime and the actual switch to the new frequency occurs immediately.

12.4.4   Low Power Bit Usage

The low-power bit (LP) is provided to allow the FLL to be disabled and conserve power when it is not being used. The DRS bits can not be written while LP bit is 1.

However, in some applications, it may be desirable to allow the FLL to be enabled and to lock for maximum accuracy before switching to an FLL engaged mode. Do this by writing the LP bit to 0.

12.4.5   DCO Maximum Frequency with 32.768 kHz Oscillator

The FLL has an option to change the clock multiplier for the selected DCO range so it results in the maximum bus frequency with a common 32.768 kHZ crystal reference clock.

12.4.6   Internal Reference Clock

When IRCLKEN is set, the internal reference clock signal is presented as ICSIRCLK, which can be used as an additional clock source. The ICSIRCLK frequency can be re-targeted by trimming the period of the internal reference clock. This can be done by writing a new value to the TRIM bits in the ICSTRM register. Writing a larger value slows down the ICSIRCLK frequency, and writing a smaller value to the ICSTRM register speeds up the ICSIRCLK frequency. The TRIM bits affect the ICSOUT frequency if the ICS is in FLL engaged internal (FEI), FLL bypassed internal (FBI), or FLL bypassed internal low-power (FBILP) mode.

Until ICSIRCLK is trimmed, programming low reference divider (RDIV) factors may result in ICSOUT frequencies that exceed the maximum chip-level frequency and violate the chip-level clock timing specifications (see the Device Overview chapter).

If IREFSTEN is set and the IRCLKEN bit is written to 1, the internal reference clock keeps running during stop mode to provide a fast recovery upon exiting stop.

All MCU devices are factory programmed with a trim value in a reserved memory location. This value is uploaded to the ICSTRM register and ICS FTRIM register during any reset initialization. For finer precision, you can trim the internal oscillator in the application and set the FTRIM bit accordingly.

12.4.7   External Reference Clock

The ICS module supports an external reference clock with frequencies between 31.25 kHz to 40 MHz in all modes. When the ERCLKEN is set, the external reference clock signal is presented as ICSERCLK, which can be used as an additional clock source. When IREFS is set, the external reference clock is not used by the FLL and is only used as ICSERCLK. In these modes, the frequency can be equal to the maximum frequency the chip-level timing specifications support (see the Device Overview chapter).

If EREFSTEN is set and the ERCLKEN bit is written to 1, the external reference clock keeps running during stop mode to provide a fast recovery upon exiting stop.

12.4.8   Fixed Frequency Clock

12.4.9   The ICS presents the divided FLL reference clock as ICSFFCLK for use as an additional clock source. ICSFFCLK frequency must be no more than 1/4 of the ICSOUT frequency to be valid.

Local Clock

The ICS presents the low range DCO output clock divided by two as ICSLCLK for use as a clock source for BDC communications. ICSLCLK is not available in FLL bypassed internal low-power (FBILP) and FLL bypassed external low-power (FBELP) modes.

Chapter 13
Inter-Integrated Circuit (S08IICV2)

13.1 Introduction

The inter-integrated circuit (IIC) provides a method of communication between a number of devices. The interface is designed to operate up to 100 kbps with maximum bus loading and timing. The device is capable of operating at higher baud rates, up to a maximum of bus clock/20, with reduced bus loading. The maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400 pF.

All MCF51QE128 Series MCUs feature the one or two IICs, as shown in Figure 13-1.

NOTE

The SDA and SCL should not be driven above V_{DD}. These pins are psuedo open-drain containing a protection diode to V_{DD}.

13.1.1 Module Configuration

The IIC1 module pins, SDA and SCL, can be repositioned under software control using SOPT2[IIC1PS] as shown in Table 13-1. This bit selects which general-purpose I/O ports are associated with IIC1 operation.

Table 13-1. IIC1 Position Options

SOPT2[IIC1PS]	Port Pin for SDA	Port Pin for SCL
0 (default)	PTA2	PTA3
1	PTB6	PTB7

13.1.2 Interrupt Vectors

For MCF51QE128 Series MCUs with two IICs, both IICs share a single interrupt vector. When interrupts are enabled for both IICs, the IICF bit must be polled in the IIC1S and IIC2S registers to determine which IIC caused the interrupt. See Chapter 8, "Interrupt Controller (CF1_INTC)," for the IIC interrupt vector assignment.

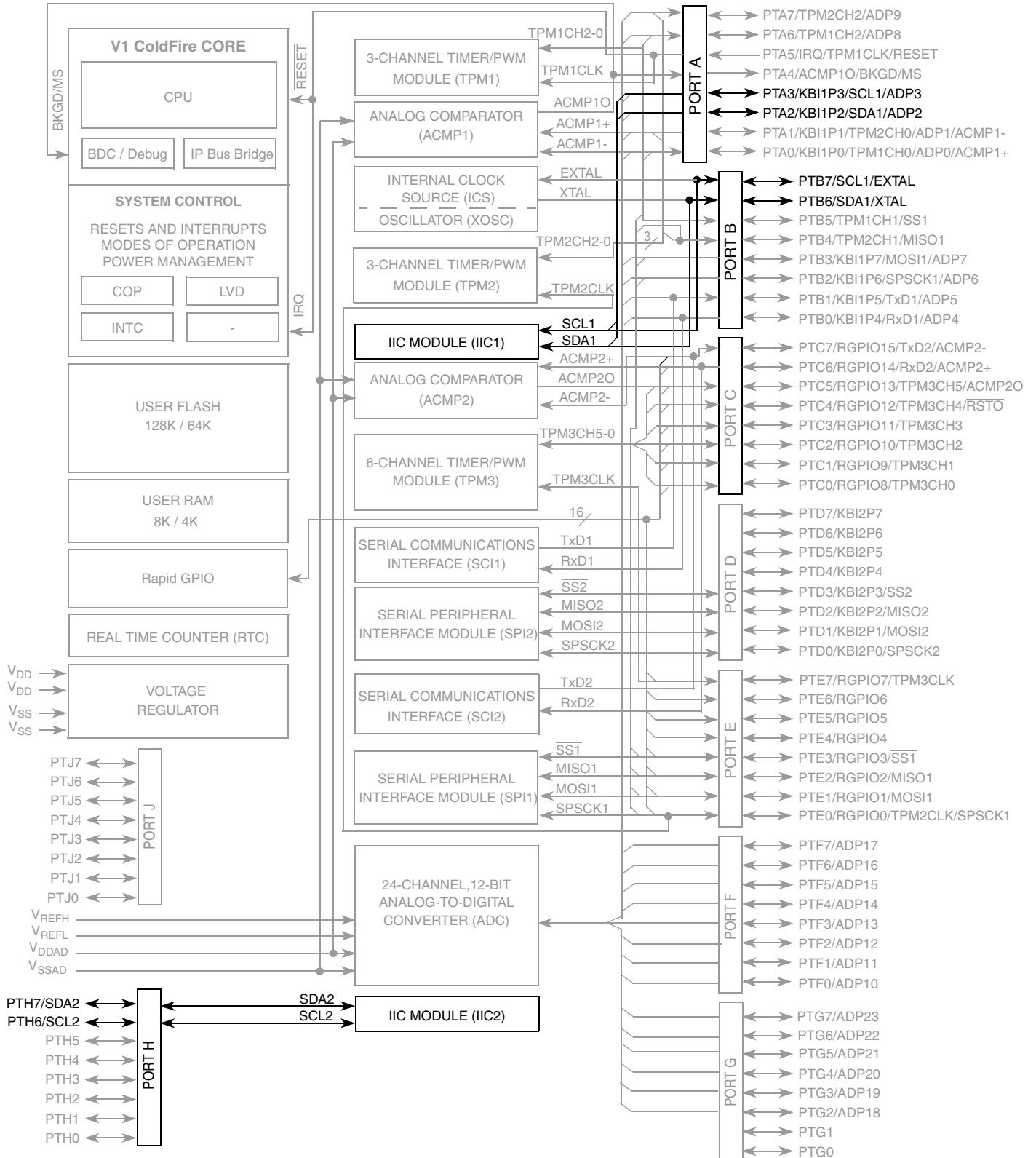**Figure 13-1. MCF51QE128 Series Block Diagram Highlighting the IIC Modules**

13.1.3 Features

The IIC includes these distinctive features:

- Compatible with IIC bus standard
- Multi-master operation
- Software programmable for one of 64 different serial clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte-by-byte data transfer
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated start signal generation
- Acknowledge bit generation/detection
- Bus busy detection
- General call recognition
- 10-bit address extension

13.1.4 Modes of Operation

A brief description of the IIC in the various MCU modes is given here.

- Run mode — This is the basic mode of operation. To conserve power in this mode, disable the module.
- Wait mode — The module continues to operate while the MCU is in wait mode and can provide a wake-up interrupt.
- Stop mode — The IIC is inactive in stop3 mode for reduced power consumption. The stop instruction does not affect IIC register states. Stop2 resets the register contents.

13.1.5 Block Diagram

Figure 13-2 is a block diagram of the IIC.
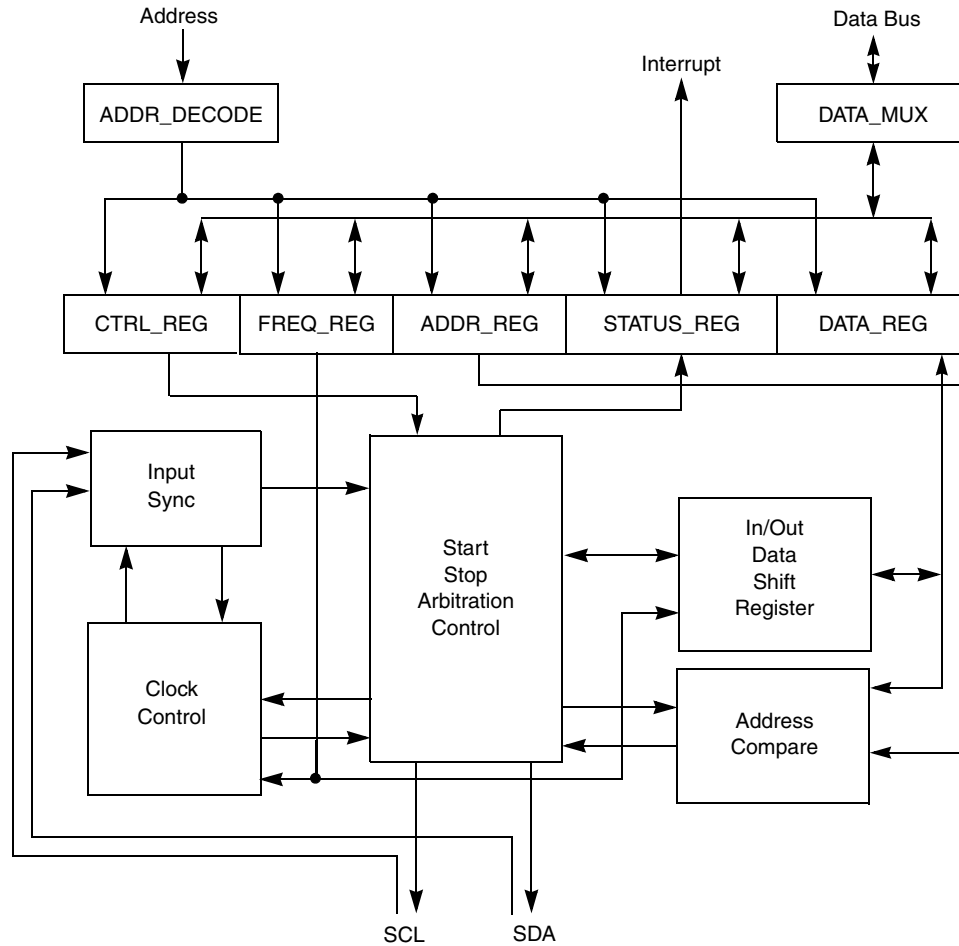
Figure 13-2. IIC Functional Block Diagram

13.2 External Signal Description

This section describes each user-accessible pin signal.

13.2.1 SCL — Serial Clock Line

The bidirectional SCL is the serial clock line of the IIC system.

13.2.2 SDA — Serial Data Line

The bidirectional SDA is the serial data line of the IIC system.

13.3 Register Definition

This section consists of the IIC register descriptions in address order.

Refer to the direct-page register summary in the memory chapter of this document for the absolute address assignments for all IIC registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

13.3.1 IIC Address Register (IICA)

	7	6	5	4	3	2	1	0
R	AD7	AD6	AD5	AD4	AD3	AD2	AD1	0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved
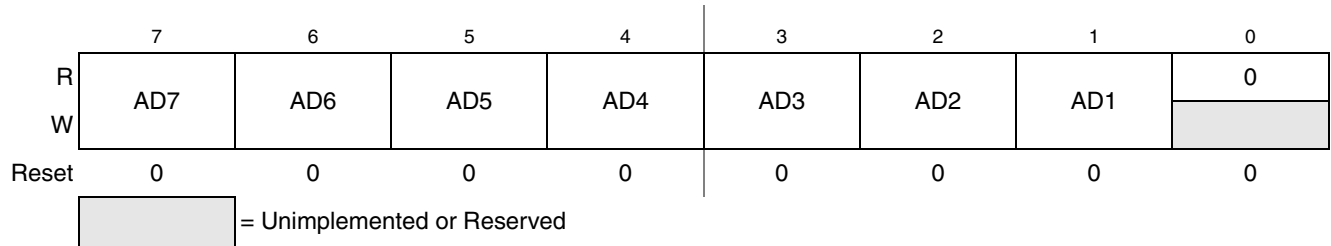
Figure 13-3. IIC Address Register (IICA)

Table 13-2. IICA Field Descriptions

Field	Description
7–1 AD[7:1]	Slave Address. The AD[7:1] field contains the slave address to be used by the IIC module. This field is used on the 7-bit address scheme and the lower seven bits of the 10-bit address scheme.

13.3.2 IIC Frequency Divider Register (IICF)

	7	6	5	4	3	2	1	0
R	MULT			ICR				
W								
Reset	0	0	0	0	0	0	0	0

Figure 13-4. IIC Frequency Divider Register (IICF)

Table 13-3. IICF Field Descriptions

Field	Description
7–6 MULT	IIC Multiplier Factor. The MULT bits define the multiplier factor, mul. This factor, along with the SCL divider, generates the IIC baud rate. The multiplier factor mul as defined by the MULT bits is provided below. 00 mul = 01 01 mul = 02 10 mul = 04 11 Reserved
5–0 ICR	IIC Clock Rate. The ICR bits are used to prescale the bus clock for bit rate selection. These bits and the MULT bits determine the IIC baud rate, the SDA hold time, the SCL Start hold time, and the SCL Stop hold time. Table 13-5 provides the SCL divider and hold values for corresponding values of the ICR. The SCL divider multiplied by multiplier factor mul generates IIC baud rate. $$\text{IIC baud rate} = \frac{\text{bus speed (Hz)}}{\text{mul} \times \text{SCLdivider}} \qquad \textbf{\textit{Eqn. 13-1}}$$ SDA hold time is the delay from the falling edge of SCL (IIC clock) to the changing of SDA (IIC data). $$\text{SDA hold time} = \text{bus period (s)} \times \text{mul} \times \text{SDA hold value} \qquad \textbf{\textit{Eqn. 13-2}}$$ SCL start hold time is the delay from the falling edge of SDA (IIC data) while SCL is high (Start condition) to the falling edge of SCL (IIC clock). $$\text{SCL Start hold time} = \text{bus period (s)} \times \text{mul} \times \text{SCL Start hold value} \qquad \textbf{\textit{Eqn. 13-3}}$$ SCL stop hold time is the delay from the rising edge of SCL (IIC clock) to the rising edge of SDA SDA (IIC data) while SCL is high (Stop condition). $$\text{SCL Stop hold time} = \text{bus period (s)} \times \text{mul} \times \text{SCL Stop hold value} \qquad \textbf{\textit{Eqn. 13-4}}$$

For example, if the bus speed is 8 MHz, the table below shows the possible hold time values with different ICR and MULT selections to achieve an IIC baud rate of 100kbps.

Table 13-4. Hold Time Values for 8 MHz Bus Speed

MULT	ICR	Hold Times (μs)		
		SDA	SCL Start	SCL Stop
0x2	0x00	3.500	4.750	5.125
0x1	0x07	2.500	4.250	5.125
0x1	0x0B	2.250	4.000	5.250
0x0	0x14	2.125	4.000	5.250
0x0	0x18	1.125	3.000	5.500

Table 13-5. IIC Divider and Hold Values

ICR (hex)	SCL Divider	SDA Hold Value	SCL Hold (Start) Value	SDA Hold (Stop) Value	ICR (hex)	SCL Divider	SDA Hold Value	SCL Hold (Start) Value	SCL Hold (Stop) Value
00	20	7	6	11	20	160	17	78	81
01	22	7	7	12	21	192	17	94	97
02	24	8	8	13	22	224	33	110	113
03	26	8	9	14	23	256	33	126	129
04	28	9	10	15	24	288	49	142	145
05	30	9	11	16	25	320	49	158	161
06	34	10	13	18	26	384	65	190	193
07	40	10	16	21	27	480	65	238	241
08	28	7	10	15	28	320	33	158	161
09	32	7	12	17	29	384	33	190	193
0A	36	9	14	19	2A	448	65	222	225
0B	40	9	16	21	2B	512	65	254	257
0C	44	11	18	23	2C	576	97	286	289
0D	48	11	20	25	2D	640	97	318	321
0E	56	13	24	29	2E	768	129	382	385
0F	68	13	30	35	2F	960	129	478	481
10	48	9	18	25	30	640	65	318	321
11	56	9	22	29	31	768	65	382	385
12	64	13	26	33	32	896	129	446	449
13	72	13	30	37	33	1024	129	510	513
14	80	17	34	41	34	1152	193	574	577
15	88	17	38	45	35	1280	193	638	641
16	104	21	46	53	36	1536	257	766	769
17	128	21	58	65	37	1920	257	958	961
18	80	9	38	41	38	1280	129	638	641
19	96	9	46	49	39	1536	129	766	769
1A	112	17	54	57	3A	1792	257	894	897
1B	128	17	62	65	3B	2048	257	1022	1025
1C	144	25	70	73	3C	2304	385	1150	1153
1D	160	25	78	81	3D	2560	385	1278	1281
1E	192	33	94	97	3E	3072	513	1534	1537
1F	240	33	118	121	3F	3840	513	1918	1921

13.3.3 IIC Control Register (IICC1)

	7	6	5	4	3	2	1	0
R	IICEN	IICIE	MST	TX	TXAK	0	0	0
W						RSTA		
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved
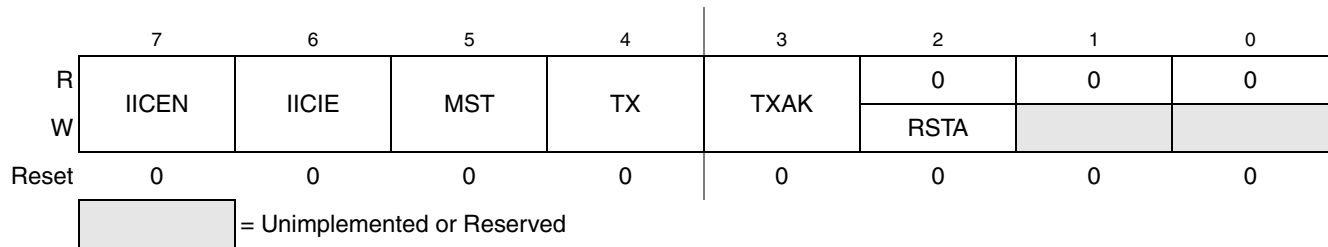
Figure 13-5. IIC Control Register (IICC1)

Table 13-6. IICC1 Field Descriptions

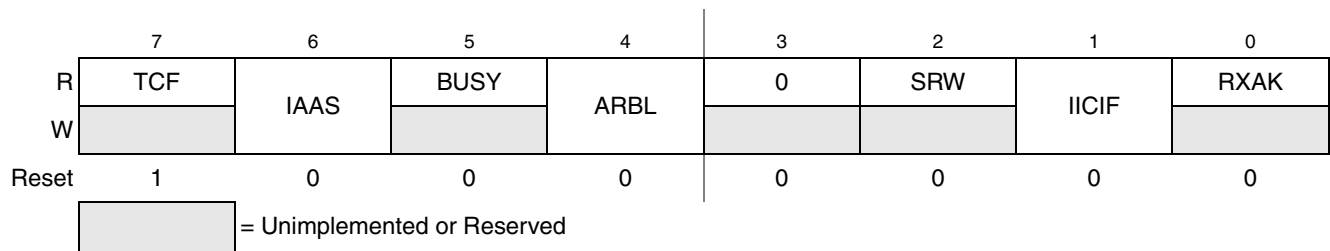| Field	Description
7 IICEN	IIC Enable. The IICEN bit determines whether the IIC module is enabled. 0 IIC is not enabled 1 IIC is enabled
6 IICIE	IIC Interrupt Enable. The IICIE bit determines whether an IIC interrupt is requested. 0 IIC interrupt request not enabled 1 IIC interrupt request enabled
5 MST	Master Mode Select. The MST bit changes from a 0 to a 1 when a start signal is generated on the bus and master mode is selected. When this bit changes from a 1 to a 0 a stop signal is generated and the mode of operation changes from master to slave. 0 Slave mode 1 Master mode
4 TX	Transmit Mode Select. The TX bit selects the direction of master and slave transfers. In master mode, this bit should be set according to the type of transfer required. Therefore, for address cycles, this bit is always high. When addressed as a slave, this bit should be set by software according to the SRW bit in the status register. 0 Receive 1 Transmit
3 TXAK	Transmit Acknowledge Enable. This bit specifies the value driven onto the SDA during data acknowledge cycles for master and slave receivers. 0 An acknowledge signal is sent out to the bus after receiving one data byte 1 No acknowledge signal response is sent
2 RSTA	Repeat start. Writing a 1 to this bit generates a repeated start condition provided it is the current master. This bit is always read as cleared. Attempting a repeat at the wrong time results in loss of arbitration.

13.3.4 IIC Status Register (IICS)

	7	6	5	4	3	2	1	0
R	TCF	IAAS	BUSY	ARBL	0	SRW	IICIF	RXAK
W								
Reset	1	0	0	0	0	0	0	0

[ ] = Unimplemented or Reserved

Figure 13-6. IIC Status Register (IICS)

Table 13-7. IICS Field Descriptions

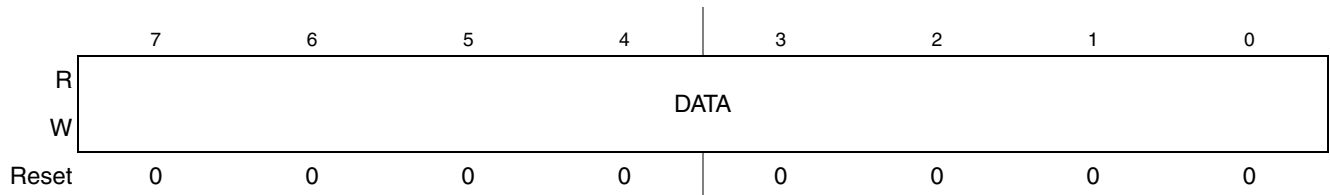| Field	Description
7 TCF	Transfer Complete Flag. This bit is set on the completion of a byte transfer. This bit is only valid during or immediately following a transfer to the IIC module or from the IIC module.The TCF bit is cleared by reading the IICD register in receive mode or writing to the IICD in transmit mode. 0 Transfer in progress 1 Transfer complete
6 IAAS	Addressed as a Slave. The IAAS bit is set when the calling address matches the programmed slave address or when the GCAEN bit is set and a general call is received. Writing the IICC register clears this bit. 0 Not addressed 1 Addressed as a slave
5 BUSY	Bus Busy. The BUSY bit indicates the status of the bus regardless of slave or master mode. The BUSY bit is set when a start signal is detected and cleared when a stop signal is detected. 0 Bus is idle 1 Bus is busy
4 ARBL	Arbitration Lost. This bit is set by hardware when the arbitration procedure is lost. The ARBL bit must be cleared by software by writing a 1 to it. 0 Standard bus operation 1 Loss of arbitration
2 SRW	Slave Read/Write. When addressed as a slave, the SRW bit indicates the value of the R/W command bit of the calling address sent to the master. 0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave
1 IICIF	IIC Interrupt Flag. The IICIF bit is set when an interrupt is pending. This bit must be cleared by software, by writing a 1 to it in the interrupt routine. One of the following events can set the IICIF bit: • One byte transfer completes • Match of slave address to calling address • Arbitration lost 0 No interrupt pending 1 Interrupt pending
0 RXAK	Receive Acknowledge. When the RXAK bit is low, it indicates an acknowledge signal has been received after the completion of one byte of data transmission on the bus. If the RXAK bit is high it means that no acknowledge signal is detected. 0 Acknowledge received 1 No acknowledge received

13.3.5 IIC Data I/O Register (IICD)

	7	6	5	4	3	2	1	0
R				DATA				
W								
Reset	0	0	0	0	0	0	0	0

Figure 13-7. IIC Data I/O Register (IICD)

Table 13-8. IICD Field Descriptions

Field	Description
7–0 DATA	**Data** — In master transmit mode, when data is written to the IICD, a data transfer is initiated. The most significant bit is sent first. In master receive mode, reading this register initiates receiving of the next byte of data.

NOTE

When transitioning out of master receive mode, the IIC mode should be switched before reading the IICD register to prevent an inadvertent initiation of a master receive data transfer.

In slave mode, the same functions are available after an address match has occurred.

The TX bit in IICC must correctly reflect the desired direction of transfer in master and slave modes for the transmission to begin. For instance, if the IIC is configured for master transmit but a master receive is desired, reading the IICD does not initiate the receive.

Reading the IICD returns the last byte received while the IIC is configured in master receive or slave receive modes. The IICD does not reflect every byte transmitted on the IIC bus, nor can software verify that a byte has been written to the IICD correctly by reading it back.

In master transmit mode, the first byte of data written to IICD following assertion of MST is used for the address transfer and should comprise of the calling address (in bit 7 to bit 1) concatenated with the required R/$\overline{\text{W}}$ bit (in position bit 0).

13.3.6 IIC Control Register 2 (IICC2)

	7	6	5	4	3	2	1	0
R	GCAEN	ADEXT	0	0	0	AD10	AD9	AD8
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved
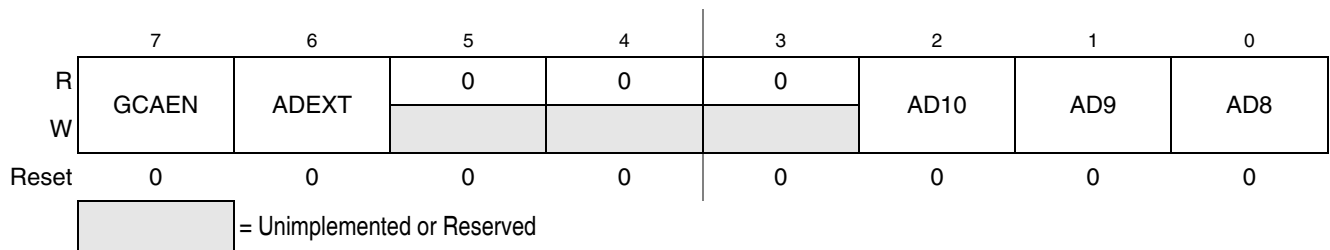
Figure 13-8. IIC Control Register (IICC2)

Table 13-9. IICC2 Field Descriptions

Field	Description
7 GCAEN	General Call Address Enable. The GCAEN bit enables or disables general call address. 0 General call address is disabled 1 General call address is enabled
6 ADEXT	Address Extension. The ADEXT bit controls the number of bits used for the slave address. 0 7-bit address scheme 1 10-bit address scheme
2–0 AD[10:8]	Slave Address. The AD[10:8] field contains the upper three bits of the slave address in the 10-bit address scheme. This field is only valid when the ADEXT bit is set.

13.4   Functional Description

This section provides a complete functional description of the IIC module.

13.4.1   IIC Protocol

The IIC bus system uses a serial data line (SDA) and a serial clock line (SCL) for data transfer. All devices connected to it must have open drain or open collector outputs. A logic AND function is exercised on both lines with external pull-up resistors. The value of these resistors is system dependent.

Normally, a standard communication is composed of four parts:

- Start signal
- Slave address transmission
- Data transfer
- Stop signal

The stop signal should not be confused with the CPU stop instruction. The IIC bus system communication is described briefly in the following sections and illustrated in Figure 13-9.
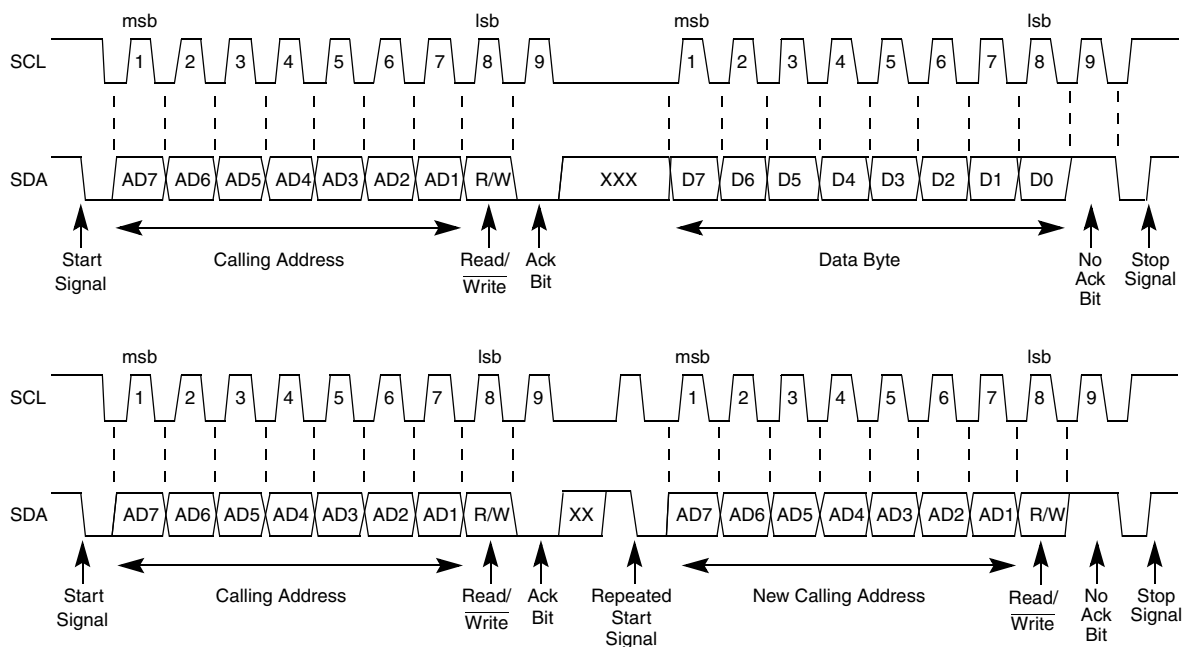
Figure 13-9. IIC Bus Transmission Signals

13.4.1.1 Start Signal

When the bus is free, no master device is engaging the bus (SCL and SDA lines are at logical high), a master may initiate communication by sending a start signal. As shown in Figure 13-9, a start signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a new data transfer (each data transfer may contain several bytes of data) and brings all slaves out of their idle states.

13.4.1.2 Slave Address Transmission

The first byte of data transferred immediately after the start signal is the slave address transmitted by the master. This is a seven-bit calling address followed by a R/$\overline{W}$ bit. The R/$\overline{W}$ bit tells the slave the desired direction of data transfer.

    1 = Read transfer, the slave transmits data to the master.
    0 = Write transfer, the master transmits data to the slave.

Only the slave with a calling address that matches the one transmitted by the master responds by sending back an acknowledge bit. This is done by pulling the SDA low at the ninth clock (see Figure 13-9).

No two slaves in the system may have the same address. If the IIC module is the master, it must not transmit an address equal to its own slave address. The IIC cannot be master and slave at the same time. However, if arbitration is lost during an address cycle, the IIC reverts to slave mode and operates correctly even if it is being addressed by another master.

13.4.1.3 Data Transfer

Before successful slave addressing is achieved, the data transfer can proceed byte-by-byte in a direction specified by the R/$\overline{\text{W}}$ bit sent by the calling master.

All transfers that come after an address cycle are referred to as data transfers, even if they carry sub-address information for the slave device

Each data byte is 8 bits long. Data may be changed only while SCL is low and must be held stable while SCL is high as shown in Figure 13-9. There is one clock pulse on SCL for each data bit, the msb being transferred first. Each data byte is followed by a 9th (acknowledge) bit, which is signalled from the receiving device. An acknowledge is signalled by pulling the SDA low at the ninth clock. In summary, one complete data transfer needs nine clock pulses.

If the slave receiver does not acknowledge the master in the ninth bit time, the SDA line must be left high by the slave. The master interprets the failed acknowledge as an unsuccessful data transfer.

If the master receiver does not acknowledge the slave transmitter after a data byte transmission, the slave interprets this as an end of data transfer and releases the SDA line.

In either case, the data transfer is aborted and the master does one of two things:

- Relinquishes the bus by generating a stop signal.
- Commences a new calling by generating a repeated start signal.

13.4.1.4 Stop Signal

The master can terminate the communication by generating a stop signal to free the bus. However, the master may generate a start signal followed by a calling command without generating a stop signal first. This is called repeated start. A stop signal is defined as a low-to-high transition of SDA while SCL at logical 1 (see Figure 13-9).

The master can generate a stop even if the slave has generated an acknowledge at which point the slave must release the bus.

13.4.1.5 Repeated Start Signal

As shown in Figure 13-9, a repeated start signal is a start signal generated without first generating a stop signal to terminate the communication. This is used by the master to communicate with another slave or with the same slave in different mode (transmit/receive mode) without releasing the bus.

13.4.1.6 Arbitration Procedure

The IIC bus is a true multi-master bus that allows more than one master to be connected on it. If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock, for which the low period is equal to the longest clock low period and the high is equal to the shortest one among the masters. The relative priority of the contending masters is determined by a data arbitration procedure, a bus master loses arbitration if it transmits logic 1 while another master transmits logic 0. The losing masters immediately switch over to slave receive mode and stop driving SDA output. In this case,

the transition from master to slave mode does not generate a stop condition. Meanwhile, a status bit is set by hardware to indicate loss of arbitration.

13.4.1.7   Clock Synchronization

Because wire-AND logic is performed on the SCL line, a high-to-low transition on the SCL line affects all the devices connected on the bus. The devices start counting their low period and after a device's clock has gone low, it holds the SCL line low until the clock high state is reached. However, the change of low to high in this device clock may not change the state of the SCL line if another device clock is still within its low period. Therefore, synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (see Figure 13-10). When all devices concerned have counted off their low period, the synchronized clock SCL line is released and pulled high. There is then no difference between the device clocks and the state of the SCL line and all the devices start counting their high periods. The first device to complete its high period pulls the SCL line low again.
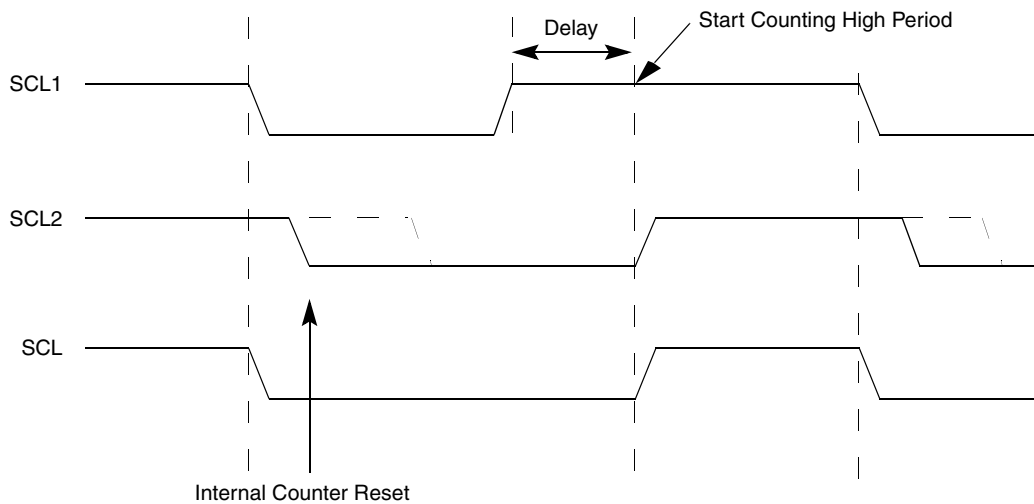

Figure 13-10. IIC Clock Synchronization

13.4.1.8   Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices may hold the SCL low after completion of one byte transfer (9 bits). In such a case, it halts the bus clock and forces the master clock into wait states until the slave releases the SCL line.

13.4.1.9   Clock Stretching

The clock synchronization mechanism can be used by slaves to slow down the bit rate of a transfer. After the master has driven SCL low the slave can drive SCL low for the required period and then release it. If the slave SCL low period is greater than the master SCL low period then the resulting SCL bus signal low period is stretched.

13.4.2 10-bit Address

For 10-bit addressing, 0x11110 is used for the first 5 bits of the first address byte. Various combinations of read/write formats are possible within a transfer that includes 10-bit addressing.

13.4.2.1 Master-Transmitter Addresses a Slave-Receiver

The transfer direction is not changed (see Table 13-10). When a 10-bit address follows a start condition, each slave compares the first seven bits of the first byte of the slave address (11110XX) with its own address and tests whether the eighth bit (R/$\overline{\text{W}}$ direction bit) is 0. More than one device can find a match and generate an acknowledge (A1). Then, each slave that finds a match compares the eight bits of the second byte of the slave address with its own address. Only one slave finds a match and generates an acknowledge (A2). The matching slave remains addressed by the master until it receives a stop condition (P) or a repeated start condition (Sr) followed by a different slave address.

S	Slave Address 1st 7 bits 11110 + AD10 + AD9	R/W 0	A1	Slave Address 2nd byte AD[8:1]	A2	Data	A	...	Data	A/A	P

Table 13-10. Master-Transmitter Addresses Slave-Receiver with a 10-bit Address

After the master-transmitter has sent the first byte of the 10-bit address, the slave-receiver sees an IIC interrupt. Software must ensure the contents of IICD are ignored and not treated as valid data for this interrupt.

13.4.2.2 Master-Receiver Addresses a Slave-Transmitter

The transfer direction is changed after the second R/$\overline{\text{W}}$ bit (see Table 13-11). Up to and including acknowledge bit A2, the procedure is the same as that described for a master-transmitter addressing a slave-receiver. After the repeated start condition (Sr), a matching slave remembers that it was addressed before. This slave then checks whether the first seven bits of the first byte of the slave address following Sr are the same as they were after the start condition (S) and tests whether the eighth (R/$\overline{\text{W}}$) bit is 1. If there is a match, the slave considers that it has been addressed as a transmitter and generates acknowledge A3. The slave-transmitter remains addressed until it receives a stop condition (P) or a repeated start condition (Sr) followed by a different slave address.

After a repeated start condition (Sr), all other slave devices also compare the first seven bits of the first byte of the slave address with their own addresses and test the eighth (R/$\overline{\text{W}}$) bit. However, none of them are addressed because R/$\overline{\text{W}}$ = 1 (for 10-bit devices) or the 11110XX slave address (for 7-bit devices) does not match.

S	Slave Address 1st 7 bits 11110 + AD10 + AD9	R/W 0	A1	Slave Address 2nd byte AD[8:1]	A2	Sr	Slave Address 1st 7 bits 11110 + AD10 + AD9	R/W 1	A3	Data	A	...	Data	A	P

Table 13-11. Master-Receiver Addresses a Slave-Transmitter with a 10-bit Address

After the master-receiver has sent the first byte of the 10-bit address, the slave-transmitter sees an IIC interrupt. Software must ensure the contents of IICD are ignored and not treated as valid data for this interrupt.

13.4.3   General Call Address

General calls can be requested in 7-bit address or 10-bit address. If the GCAEN bit is set, the IIC matches the general call address as well as its own slave address. When the IIC responds to a general call, it acts as a slave-receiver and the IAAS bit is set after the address cycle. Software must read the IICD register after the first byte transfer to determine whether the address matches is its own slave address or a general call. If the value is 00, the match is a general call. If the GCAEN bit is clear, the IIC ignores any data supplied from a general call address by not issuing an acknowledgement.

13.5   Resets

The IIC is disabled after reset. The IIC cannot cause an MCU reset.

13.6   Interrupts

The IIC generates a single interrupt.

An interrupt from the IIC is generated when any of the events in Table 13-12 occur, provided the IICIE bit is set. The interrupt is driven by bit IICIF (of the IIC status register) and masked with bit IICIE (of the IIC control register). The IICIF bit must be cleared by software by writing a 1 to it in the interrupt routine. You can determine the interrupt type by reading the status register.

Table 13-12. Interrupt Summary

Interrupt Source	Status	Flag	Local Enable
Complete 1-byte transfer	TCF	IICIF	IICIE
Match of received calling address	IAAS	IICIF	IICIE
Arbitration Lost	ARBL	IICIF	IICIE

13.6.1   Byte Transfer Interrupt

The TCF (transfer complete flag) bit is set at the falling edge of the ninth clock to indicate the completion of byte transfer.

13.6.2   Address Detect Interrupt

When the calling address matches the programmed slave address (IIC address register) or when the GCAEN bit is set and a general call is received, the IAAS bit in the status register is set. The CPU is interrupted, provided the IICIE is set. The CPU must check the SRW bit and set its Tx mode accordingly.

13.6.3   Arbitration Lost Interrupt

The IIC is a true multi-master bus that allows more than one master to be connected on it. If two or more masters try to control the bus at the same time, the relative priority of the contending masters is determined by a data arbitration procedure. The IIC module asserts this interrupt when it loses the data arbitration process and the ARBL bit in the status register is set.

Arbitration is lost in the following circumstances:

- SDA sampled as a low when the master drives a high during an address or data transmit cycle.
- SDA sampled as a low when the master drives a high during the acknowledge bit of a data receive cycle.
- A start cycle is attempted when the bus is busy.
- A repeated start cycle is requested in slave mode.
- A stop condition is detected when the master did not request it.

This bit must be cleared by software writing a 1 to it.

13.7 Initialization/Application Information

Module Initialization (Slave)
1. Write: IICC2
   — to enable or disable general call
   — to select 10-bit or 7-bit addressing mode
2. Write: IICA
   — to set the slave address
3. Write: IICC1
   — to enable IIC and interrupts
4. Initialize RAM variables (IICEN = 1 and IICIE = 1) for transmit data
5. Initialize RAM variables used to achieve the routine shown in Figure 13-12

Module Initialization (Master)
1. Write: IICF
   — to set the IIC baud rate (example provided in this chapter)
2. Write: IICC1
   — to enable IIC and interrupts
3. Initialize RAM variables (IICEN = 1 and IICIE = 1) for transmit data
4. Initialize RAM variables used to achieve the routine shown in Figure 13-12
5. Write: IICC1
   — to enable TX
6. Write: IICC1
   — to enable MST (master mode)
7. Write: IICD
   — with the address of the target slave. (The lsb of this byte determines whether the communication is master receive or transmit.)

Module Use

The routine shown in Figure 13-12 can handle both master and slave IIC operations. For slave operation, an incoming IIC message that contains the proper address begins IIC communication. For master operation, communication must be initiated by writing to the IICD register.

Register Model

IICA	AD[7:1]						0

When addressed as a slave (in slave mode), the module responds to this address

IICF	MULT		ICR				

Baud rate = BUSCLK / (2 x MULT x (SCL DIVIDER))

IICC1	IICEN	IICIE	MST	TX	TXAK	RSTA	0	0

Module configuration

IICS	TCF	IAAS	BUSY	ARBL	0	SRW	IICIF	RXAK

Module status flags

IICD	DATA						

Data register; Write to transmit IIC data read to read IIC data

IICC2	GCAEN	ADEXT	0	0	0	AD10	AD9	AD8

Address configuration

Figure 13-11. IIC Module Quick Start

Figure 13-12. Typical IIC Interrupt Routine
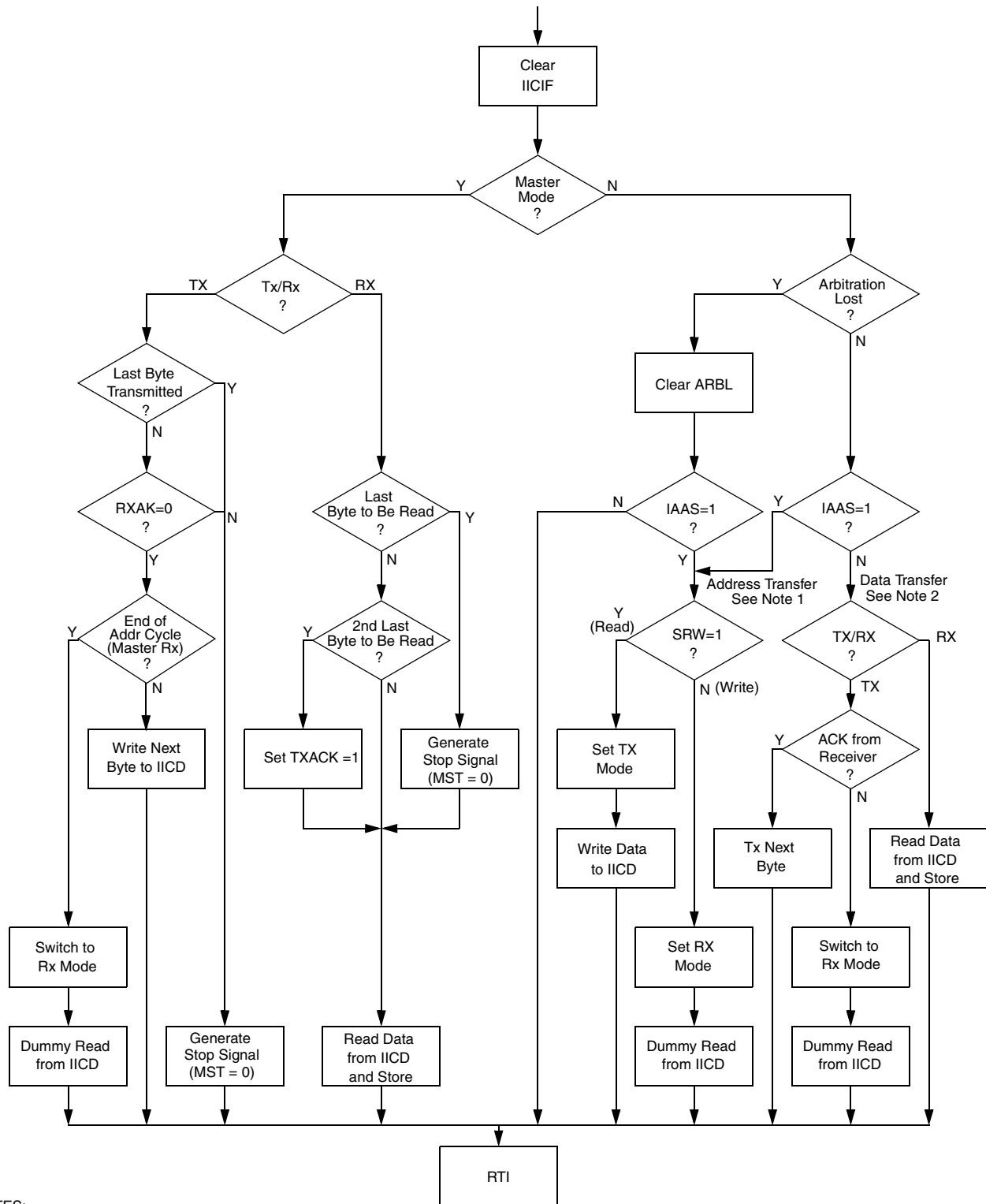
NOTES:
1. If general call is enabled, a check must be done to determine whether the received address was a general call address (0x00). If the received address was a general call address, then the general call must be handled by user software.
2. When 10-bit addressing is used to address a slave, the slave sees an interrupt following the first byte of the extended address. User software must ensure that for this interrupt, the contents of IICD are ignored and not treated as a valid data transfer

Chapter 14
Real-Time Counter (S08RTCV1)

14.1 Introduction

The real-time counter (RTC) consists of one 8-bit counter, one 8-bit comparator, several binary-based and decimal-based prescaler dividers, three clock sources, and one programmable periodic interrupt. This module can be used for time-of-day, calendar, or any task scheduling functions. It can also serve as a cyclic wake up from low-power modes without the need of external components.

14.1.1 ADC Hardware Trigger

The RTC can be enabled as a hardware trigger for the ADC module by setting ADCSC2[ADTRG]. When enabled, the ADC is triggered each time RTCINT matches RTCMOD. The RTC interrupt does not have to be enabled to trigger the ADC.

14.1.2 RTC Clock Sources

The RTC module on MCF51QE128 Series can be clocked from ICSIRCLK, OSCOUT, or the LPO.

In this chapter, ERCLK is replaced by OSCOUT for this MCU.

14.1.3 RTC Modes of Operation

All clock sources are available in all modes except stop2. The OSCOUT and LPO can be enabled as the clock source of the RTC in stop2.

14.1.3.1 RTC Status after Stop2 Wakeup

The registers associated with the RTC are unaffected after a stop2 wakeup.

14.1.3.2 Clocks in Stop Modes

In the MCF51QE128 Series, LPO and OSCOUT can be used in stop2 and stop3. IRCLK is available only in stop3.

14.1.4 RTC Clock Gating

The bus clock to the RTC can be gated on and off with SCGC2[RTC]. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the RTC bit can be cleared to disable the clock to this module when not in use. See Section 5.6, "Peripheral Clock Gating," for details.

14.1.5 Interrupt Vector

See Chapter 8, "Interrupt Controller (CF1_INTC)," for the RTC interrupt vector assignment.
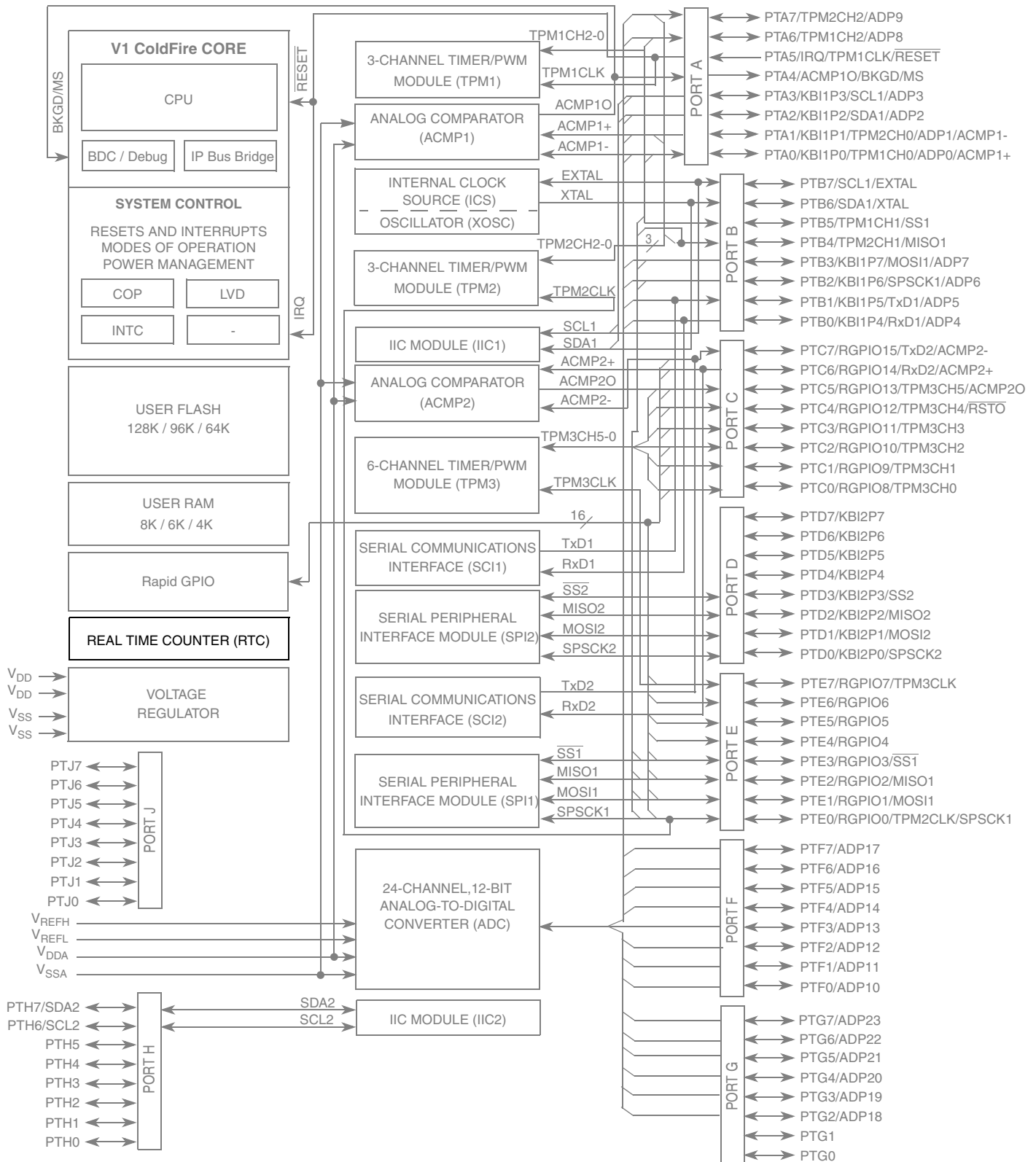
MCF51QE128 Block Diagram Highlighting RTC Block and Pins

14.1.6   Features

Features of the RTC module include:

- 8-bit up-counter
  - — 8-bit modulo match limit
  - — Software controllable periodic interrupt on match
- Three software selectable clock sources for input to prescaler with selectable binary-based and decimal-based divider values
  - — 1-kHz internal low-power oscillator (LPO)
  - — External clock (ERCLK)
  - — 32-kHz internal clock (IRCLK)

14.1.7   Modes of Operation

This section defines the operation in stop, wait and background debug modes.

14.1.7.1   Wait Mode

The RTC continues to run in wait mode if enabled before executing the appropriate instruction. Therefore, the RTC can bring the MCU out of wait mode if the real-time interrupt is enabled. For lowest possible current consumption, the RTC should be stopped by software if not needed as an interrupt source during wait mode.

14.1.7.2   Stop Modes

The RTC continues to run in stop2 or stop3 mode if the RTC is enabled before executing the STOP instruction. Therefore, the RTC can bring the MCU out of stop modes with no external components, if the real-time interrupt is enabled.

The LPO clock can be used in stop2 and stop3 modes. ERCLK and IRCLK clocks are only available in stop3 mode.

Power consumption is lower when all clock sources are disabled, but in that case, the real-time interrupt cannot wake up the MCU from stop modes.

14.1.7.3   Active Background Mode

The RTC suspends all counting during active background mode until the microcontroller returns to normal user operating mode. Counting resumes from the suspended value as long as the RTCMOD register is not written and the RTCPS and RTCLKS bits are not altered.

14.1.8   Block Diagram

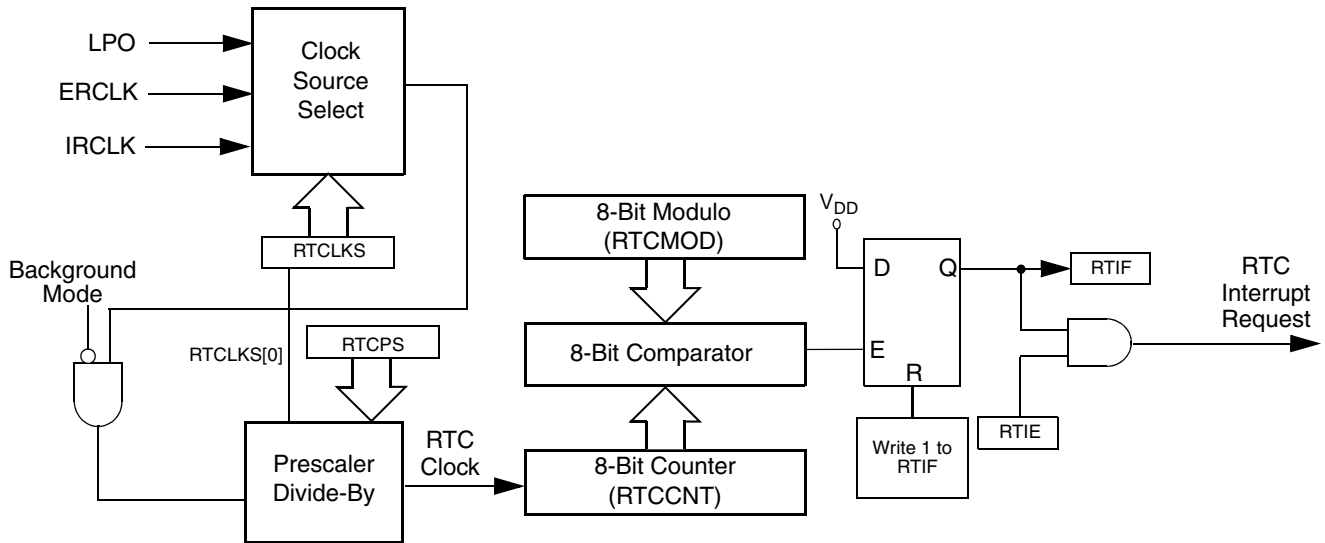The block diagram for the RTC module is shown in Figure 14-1.

Figure 14-1. Real-Time Counter (RTC) Block Diagram

14.2   External Signal Description

The RTC does not include any off-chip signals.

14.3   Register Definition

The RTC includes a status and control register, an 8-bit counter register, and an 8-bit modulo register.

Refer to the direct-page register summary in the memory section of this document for the absolute address assignments for all RTC registers.This section refers to registers and control bits only by their names and relative address offsets.

Table 14-1 is a summary of RTC registers.

Table 14-1. RTC Register Summary

Name		7	6	5	4	3	2	1	0
RTCSC	R	RTIF	RTCLKS		RTIE	RTCPS			
	W	RTIF	RTCLKS		RTIE	RTCPS			
RTCCNT	R	RTCCNT							
	W								
RTCMOD	R	RTCMOD							
	W	RTCMOD							

14.3.1　RTC Status and Control Register (RTCSC)

RTCSC contains the real-time interrupt status flag (RTIF), the clock select bits (RTCLKS), the real-time interrupt enable bit (RTIE), and the prescaler select bits (RTCPS).
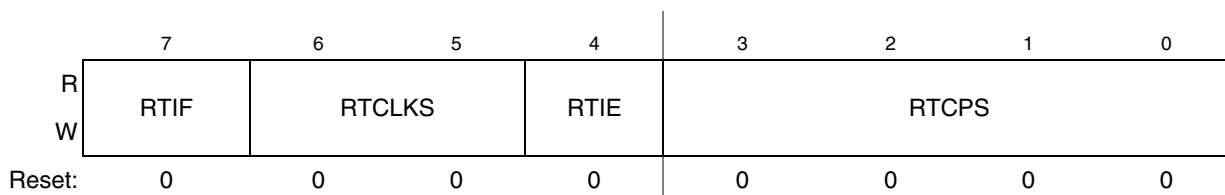
	7	6	5	4	3	2	1	0
R	RTIF	RTCLKS		RTIE	RTCPS			
W								
Reset:	0	0	0	0	0	0	0	0

Figure 14-2. RTC Status and Control Register (RTCSC)

Table 14-2. RTCSC Field Descriptions

Field	Description
7 RTIF	Real-Time Interrupt Flag This status bit indicates the RTC counter register reached the value in the RTC modulo register. Writing a logic 0 has no effect. Writing a logic 1 clears the bit and the real-time interrupt request. Reset clears RTIF. 0　RTC counter has not reached the value in the RTC modulo register. 1　RTC counter has reached the value in the RTC modulo register.
6–5 RTCLKS	Real-Time Clock Source Select. These two read/write bits select the clock source input to the RTC prescaler. Changing the clock source clears the prescaler and RTCCNT counters. When selecting a clock source, ensure that the clock source is properly enabled (if applicable) to ensure correct operation of the RTC. Reset clears RTCLKS. 00 Real-time clock source is the 1-kHz low power oscillator (LPO) 01 Real-time clock source is the external clock (ERCLK) 1x Real-time clock source is the internal clock (IRCLK)
4 RTIE	Real-Time Interrupt Enable. This read/write bit enables real-time interrupts. If RTIE is set, then an interrupt is generated when RTIF is set. Reset clears RTIE. 0　Real-time interrupt requests are disabled. Use software polling. 1　Real-time interrupt requests are enabled.
3–0 RTCPS	Real-Time Clock Prescaler Select. These four read/write bits select binary-based or decimal-based divide-by values for the clock source. See Table 14-3. Changing the prescaler value clears the prescaler and RTCCNT counters. Reset clears RTCPS.

Table 14-3. RTC Prescaler Divide-by values

RTCLKS[0]	RTCPS															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Off	2^3	2^5	2^6	2^7	2^8	2^9	2^{10}	1	2	2^2	10	2^4	10^2	5×10^2	10^3
1	Off	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	10^3	2×10^3	5×10^3	10^4	2×10^4	5×10^4	10^5	2×10^5

14.3.2   RTC Counter Register (RTCCNT)

RTCCNT is the read-only value of the current RTC count of the 8-bit counter.

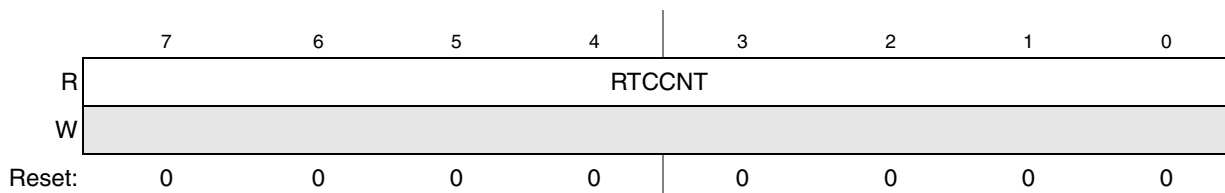|	7	6	5	4	3	2	1	0
R				RTCCNT				
W								
Reset:	0	0	0	0	0	0	0	0

Figure 14-3. RTC Counter Register (RTCCNT)

Table 14-4. RTCCNT Field Descriptions

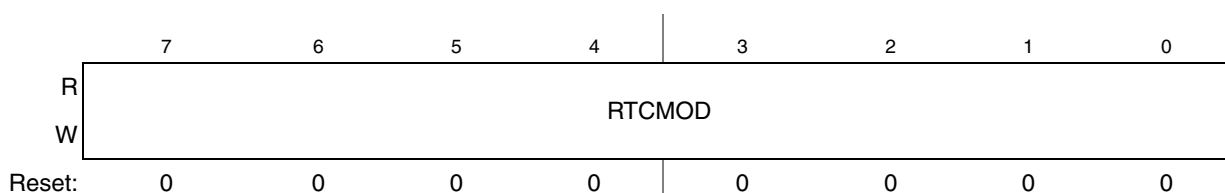| Field	Description
7:0 RTCCNT	RTC Count. These eight read-only bits contain the current value of the 8-bit counter. Writes have no effect to this register. Reset, writing to RTCMOD, or writing different values to RTCLKS and RTCPS clear the count to 0x00.

14.3.3   RTC Modulo Register (RTCMOD)

	7	6	5	4	3	2	1	0
R				RTCMOD				
W								
Reset:	0	0	0	0	0	0	0	0

Figure 14-4. RTC Modulo Register (RTCMOD)

Table 14-5. RTCMOD Field Descriptions

Field	Description
7:0 RTCMOD	RTC Modulo. These eight read/write bits contain the modulo value used to reset the count to 0x00 upon a compare match and set the RTIF status bit. A value of 0x00 sets the RTIF bit on each rising edge of the prescaler output. Writing to RTCMOD resets the prescaler and the RTCCNT counters to 0x00. Reset sets the modulo to 0x00.

14.4   Functional Description

The RTC is composed of a main 8-bit up-counter with an 8-bit modulo register, a clock source selector, and a prescaler block with binary-based and decimal-based selectable values. The module also contains software selectable interrupt logic.

After any MCU reset, the counter is stopped and reset to 0x00, the modulus register is set to 0x00, and the prescaler is off. The 1-kHz internal oscillator clock is selected as the default clock source. To start the prescaler, write any value other than zero to the prescaler select bits (RTCPS).

Three clock sources are software selectable: the low power oscillator clock (LPO), the external clock (ERCLK), and the internal clock (IRCLK). The RTC clock select bits (RTCLKS) select the desired clock source. If a different value is written to RTCLKS, the prescaler and RTCCNT counters are reset to 0x00.

RTCPS and the RTCLKS[0] bit select the desired divide-by value. If a different value is written to RTCPS, the prescaler and RTCCNT counters are reset to 0x00. Table 14-6 shows different prescaler period values.

Table 14-6. Prescaler Period

RTCPS	1-kHz Internal Clock (RTCLKS = 00)	1-MHz External Clock (RTCLKS = 01)	32-kHz Internal Clock (RTCLKS = 10)	32-kHz Internal Clock (RTCLKS = 11)
0000	Off	Off	Off	Off
0001	8 ms	1.024 ms	250 μs	32 ms
0010	32 ms	2.048 ms	1 ms	64 ms
0011	64 ms	4.096 ms	2 ms	128 ms
0100	128 ms	8.192 ms	4 ms	256 ms
0101	256 ms	16.4 ms	8 ms	512 ms
0110	512 ms	32.8 ms	16 ms	1.024 s
0111	1.024 s	65.5 ms	32 ms	2.048 s
1000	1 ms	1 ms	31.25 μs	31.25 ms
1001	2 ms	2 ms	62.5 μs	62.5 ms
1010	4 ms	5 ms	125 μs	156.25 ms
1011	10 ms	10 ms	312.5 μs	312.5 ms
1100	16 ms	20 ms	0.5 ms	0.625 s
1101	0.1 s	50 ms	3.125 ms	1.5625 s
1110	0.5 s	0.1 s	15.625 ms	3.125 s
1111	1 s	0.2 s	31.25 ms	6.25 s

The RTC modulo register (RTCMOD) allows the compare value to be set to any value from 0x00 to 0xFF. When the counter is active, the counter increments at the selected rate until the count matches the modulo value. When these values match, the counter resets to 0x00 and continues counting. The real-time interrupt flag (RTIF) is set when a match occurs. The flag sets on the transition from the modulo value to 0x00. Writing to RTCMOD resets the prescaler and the RTCCNT counters to 0x00.

The RTC allows for an interrupt to be generated when RTIF is set. To enable the real-time interrupt, set the real-time interrupt enable bit (RTIE) in RTCSC. RTIF is cleared by writing a 1 to RTIF.

14.4.1 RTC Operation Example

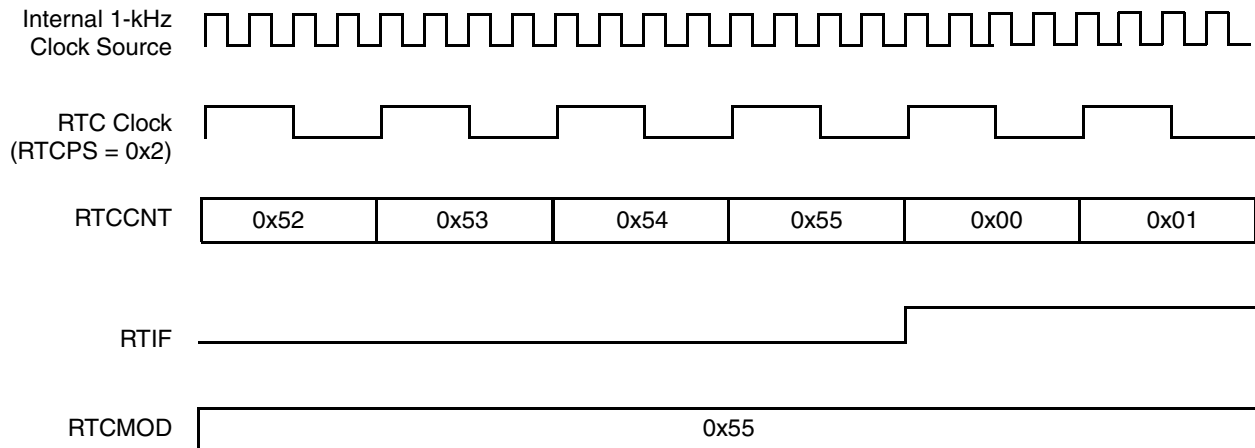This section shows an example of the RTC operation as the counter reaches a matching value from the modulo register.

Figure 14-5. RTC Counter Overflow Example

In the example of Figure 14-5, the selected clock source is the internal clock source. The prescaler (RTCPS) is set to 0x2 or divide-by-4. The modulo value in the RTCMOD register is set to 0x55. When the counter, RTCCNT, reaches the modulo value of 0x55, the counter overflows to 0x00 and continues counting. The real-time interrupt flag, RTIF, sets when the counter value changes from 0x55 to 0x00. A real-time interrupt is generated when RTIF is set, if RTIE is set.

'b00the clock ofthe clock of flip-flop is

14.5   Initialization/Application Information

This section provides example code to give some basic direction to a user on how to initialize and configure the RTC module. The example software is implemented in C language.

The example below shows how to implement time of day with the RTC using the 1-kHz clock source to achieve the lowest possible power consumption. Because the 1-kHz clock source is not as accurate as a crystal, software can be added for any adjustments. For accuracy without adjustments at the expense of additional power consumption, the external clock (ERCLK) or the internal clock (IRCLK) can be selected with appropriate prescaler and modulo values.

```
/* Initialize the elapsed time counters */
Seconds = 0;
Minutes = 0;
Hours = 0;
Days=0;


/* Configure RTC to interrupt every 1 second from 1-kHz clock source */
RTCMOD.byte = 0x00;
RTCSC.byte = 0x1F;


/*****************************************************************
Function Name : RTC_ISR
```

```
Notes : Interrupt service routine for RTC module.
*********************************************************************/
#pragma TRAP_PROC
void RTC_ISR(void)
{
        /* Clear the interrupt flag */
        RTCSC.byte = RTCSC.byte | 0x80;
        /* RTC interrupts every 1 Second */
        Seconds++;
        /* 60 seconds in a minute */
        if (Seconds > 59){
        Minutes++;
        Seconds = 0;
        }
        /* 60 minutes in an hour */
        if (Minutes > 59){
        Hours++;
        Minutes = 0;
        }
        /* 24 hours in a day */
        if (Hours > 23){
        Days ++;
        Hours = 0;
        }
```

Chapter 15
Serial Communications Interface (S08SCIV4)

15.1 Introduction

Figure 15-1 shows the MCF51QE128 Series block diagram with the SCI highlighted.

NOTE

Ignore any references to stop1 low-power mode in this chapter, because the MCF51QE128 device does not support it.

15.1.1 SCI Clock Gating

The bus clock to SCI1 and SCI2 can be gated on and off using the SCGC1[SCI1,SCI2] bits, respectively. These bits are set after any reset, which enables the bus clock to these modules. To conserve power, these bits can be cleared to disable the clock to either of these modules when not in use. See Section 5.6, "Peripheral Clock Gating," for details.

15.1.2 Interrupt Vectors

Each SCI module contains three interrupt sources: transmit, receive, and error. See Chapter 8, "Interrupt Controller (CF1_INTC)," for a list of the SCI interrupt vector assignments.
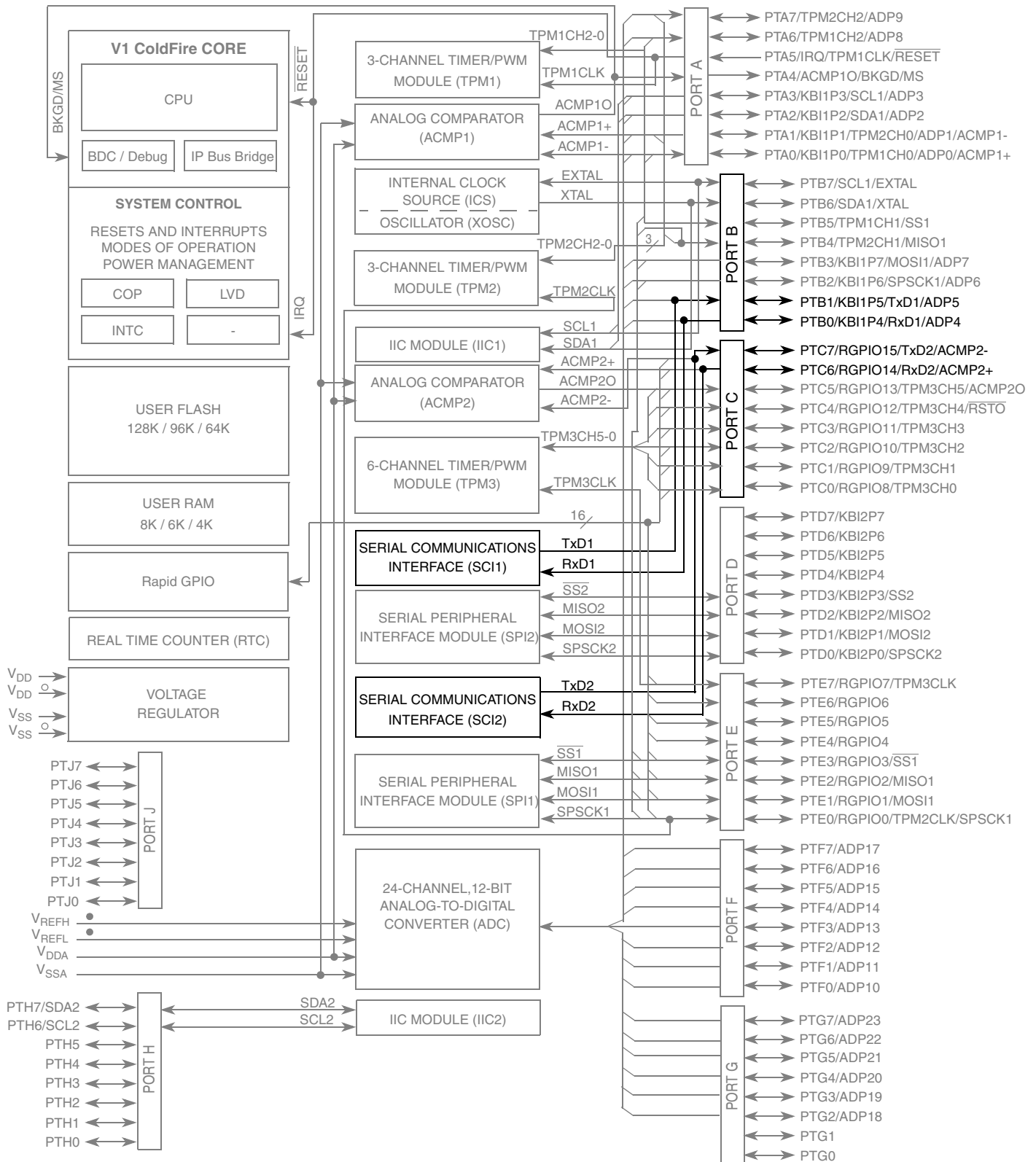
Figure 15-1. MCF51QE128 Series Block Diagram Highlighting SCI Block and Pins

Module Initialization:

Write: SCIxBDH:SCIxBDL   to set   baud rate

Write: SCIxC1   to configure   1-wire/2-wire, 9/8-bit data, wakeup, and parity, if used.

Write;   SCIxC2   to configure   interrupts, enable Rx and Tx, RWU
                                          Enable Rx wakeup, SBK sends break character

Write: SCIxC3   to enable   Rx error interrupt sources. Also controls pin direction in
                                       1-wire modes. R8 and T8 only used in 9-bit data modes.

Module Use:

Wait for TDRE, then write data to SCIxD

Wait for RDRF, then read data from SCIxD

A small number of applications will use RWU to manage automatic receiver wakeup, SBK to send break characters, and R8 and T8 for 9-bit data.
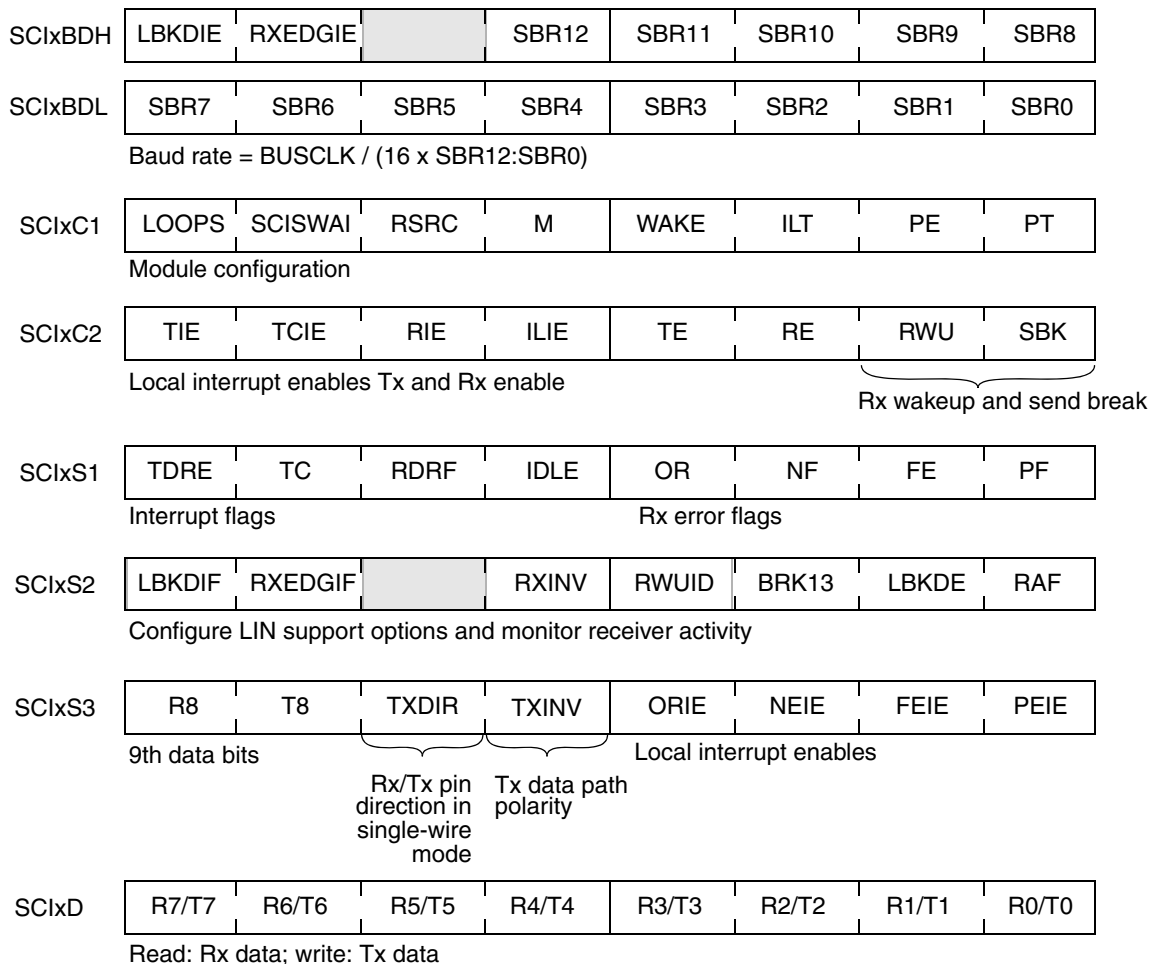
SCIxBDH	LBKDIE	RXEDGIE		SBR12	SBR11	SBR10	SBR9	SBR8

SCIxBDL	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0

Baud rate = BUSCLK / (16 x SBR12:SBR0)

SCIxC1	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT

Module configuration

SCIxC2	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK

Local interrupt enables Tx and Rx enable   Rx wakeup and send break

SCIxS1	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF

Interrupt flags   Rx error flags

SCIxS2	LBKDIF	RXEDGIF		RXINV	RWUID	BRK13	LBKDE	RAF

Configure LIN support options and monitor receiver activity

SCIxS3	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE

9th data bits   Local interrupt enables

Rx/Tx pin   Tx data path
direction in   polarity
single-wire
mode

SCIxD	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0

Read: Rx data; write: Tx data

Figure 15-2. SCI Module Quick Start

MCF51QE128 MCU Series Reference Manual, Rev. 3

15.1.3   Features

Features of SCI module include:

- Full-duplex, standard non-return-to-zero (NRZ) format
- Double-buffered transmitter and receiver with separate enables
- Programmable baud rates (13-bit modulo divider)
- Interrupt-driven or polled operation:
  — Transmit data register empty and transmission complete
  — Receive data register full
  — Receive overrun, parity error, framing error, and noise error
  — Idle receiver detect
  — Active edge on receive pin
  — Break detect supporting LIN
- Hardware parity generation and checking
- Programmable 8-bit or 9-bit character length
- Receiver wakeup by idle-line or address-mark
- Optional 13-bit break character generation / 11-bit break character detection
- Selectable transmitter output polarity

15.1.4   Modes of Operation

See Section 15.3, "Functional Description," for details concerning SCI operation in these modes:

- 8- and 9-bit data modes
- Stop mode operation
- Loop mode
- Single-wire mode

15.1.5   Block Diagram

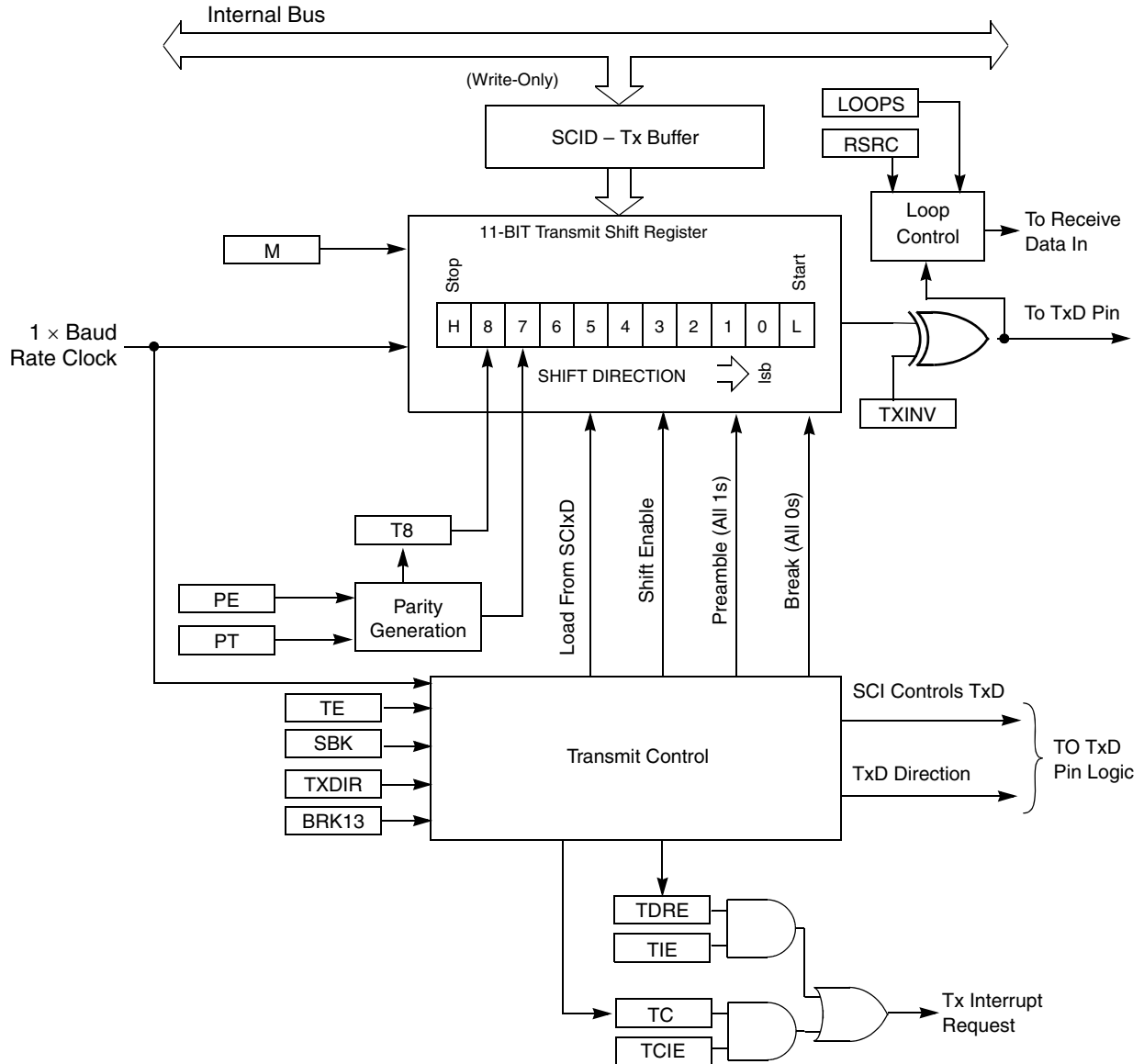Figure 15-3 shows the transmitter portion of the SCI.

Figure 15-3. SCI Transmitter Block Diagram

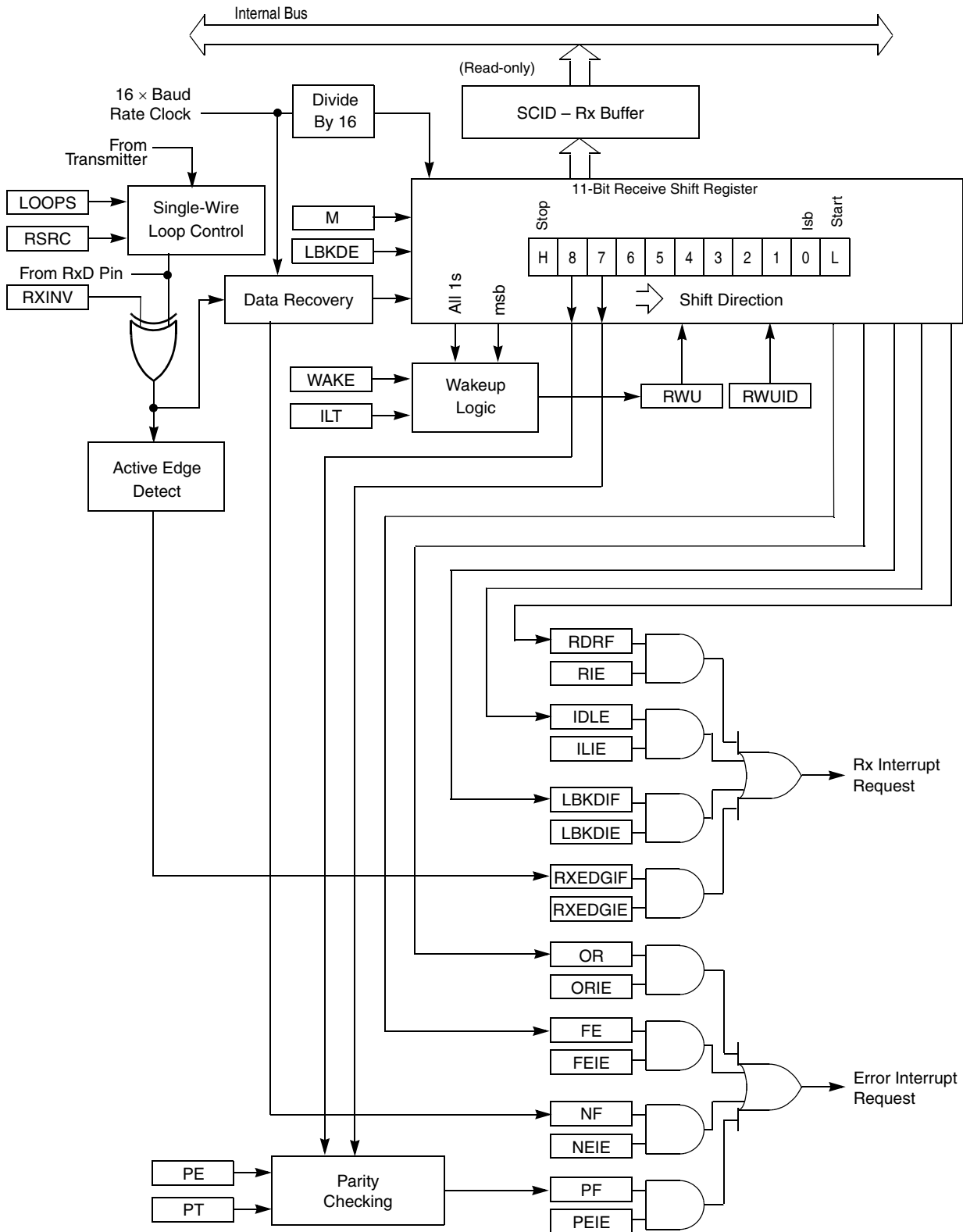Figure 15-4 shows the receiver portion of the SCI.

Figure 15-4. SCI Receiver Block Diagram

MCF51QE128 MCU Series Reference Manual, Rev. 3

15.2 Register Definition

The SCI has eight 8-bit registers to control baud rate, select SCI options, report SCI status, and for transmit/receive data.

Refer to the direct-page register summary in the memory chapter of this document or the absolute address assignments for all SCI registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

15.2.1 SCI Baud Rate Registers (SCIxBDH, SCIxBDL)

This pair of registers controls the prescale divisor for SCI baud rate generation. To update the 13-bit baud rate setting [SBR12:SBR0], first write to SCIxBDH to buffer the high half of the new value and then write to SCIxBDL. The working value in SCIxBDH does not change until SCIxBDL is written.

SCIxBDL is reset to a non-zero value, so after reset the baud rate generator remains disabled until the first time the receiver or transmitter is enabled (RE or TE bits in SCIxC2 are written to 1).
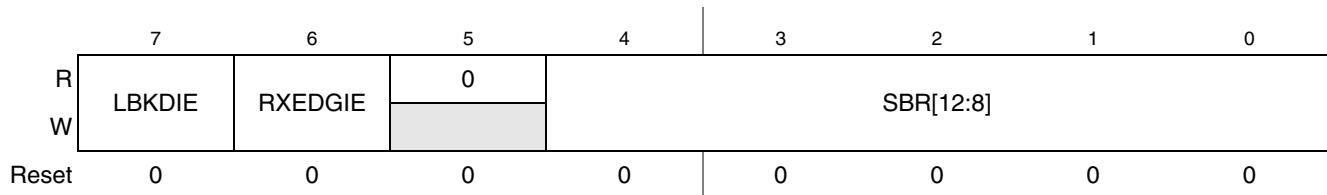
	7	6	5	4	3	2	1	0
R	LBKDIE	RXEDGIE	0		SBR[12:8]			
W	LBKDIE	RXEDGIE			SBR[12:8]			
Reset	0	0	0	0	0	0	0	0

Figure 15-5. SCI Baud Rate Register (SCIxBDH)

Table 15-1. SCIxBDH Field Descriptions

Field	Description
7 LBKDIE	LIN Break Detect Interrupt Enable (for LBKDIF) 0 Hardware interrupts from LBKDIF disabled (use polling). 1 Hardware interrupt requested when LBKDIF flag is 1.
6 RXEDGIE	RxD Input Active Edge Interrupt Enable (for RXEDGIF) 0 Hardware interrupts from RXEDGIF disabled (use polling). 1 Hardware interrupt requested when RXEDGIF flag is 1.
4–0 SBR[12:8]	Baud Rate Modulo Divisor. The 13 bits in SBR[12:0] are referred to collectively as BR, and they set the modulo divide rate for the SCI baud rate generator. When BR is cleared, the SCI baud rate generator is disabled to reduce supply current. When BR is 1 – 8191, the SCI baud rate equals BUSCLK/(16×BR). See also BR bits in Table 15-2.

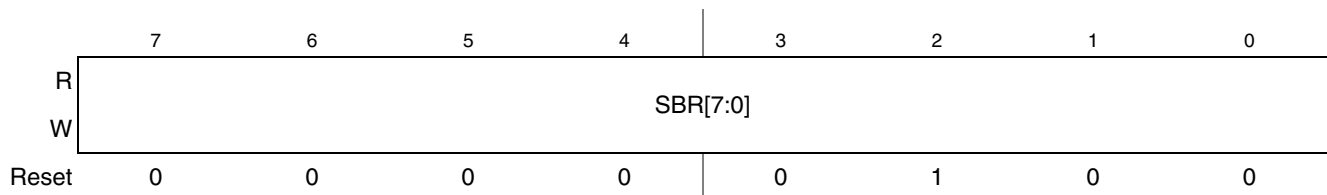|	7	6	5	4	3	2	1	0
R				SBR[7:0]				
W				SBR[7:0]				
Reset	0	0	0	0	0	1	0	0

Figure 15-6. SCI Baud Rate Register (SCIxBDL)

Table 15-2. SCIxBDL Field Descriptions

Field	Description
7–0 SBR[7:0]	Baud Rate Modulo Divisor. These 13 bits in SBR[12:0] are referred to collectively as BR, and they set the modulo divide rate for the SCI baud rate generator. When BR is cleared, the SCI baud rate generator is disabled to reduce supply current. When BR is 1 – 8191, the SCI baud rate equals BUSCLK/(16×BR). See also BR bits in Table 15-1.

15.2.2   SCI Control Register 1 (SCIxC1)

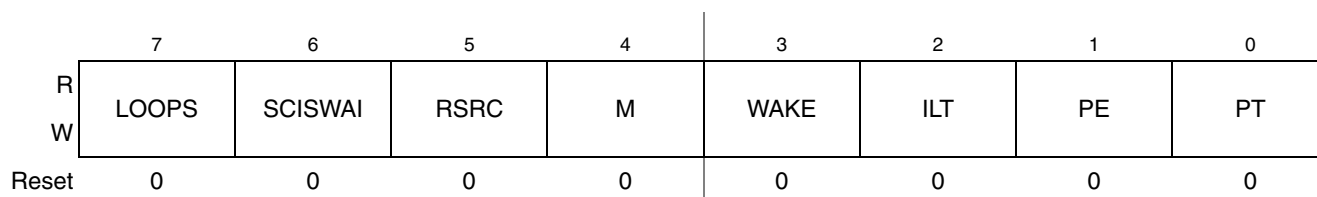This read/write register controls various optional features of the SCI system.

	7	6	5	4	3	2	1	0
R W	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
Reset	0	0	0	0	0	0	0	0

Figure 15-7. SCI Control Register 1 (SCIxC1)

Table 15-3. SCIxC1 Field Descriptions

Field	Description
7 LOOPS	Loop Mode Select. Selects between loop back modes and normal 2-pin full-duplex modes. When LOOPS is set, the transmitter output is internally connected to the receiver input. 0 Normal operation — RxD and TxD use separate pins. 1 Loop mode or single-wire mode where transmitter outputs are internally connected to receiver input. (See RSRC bit.) RxD pin is not used by SCI.
6 SCISWAI	SCI Stops in Wait Mode 0 SCI clocks continue to run in wait mode so the SCI can be the source of an interrupt that wakes up the CPU. 1 SCI clocks freeze while CPU is in wait mode.
5 RSRC	Receiver Source Select. This bit has no meaning or effect unless the LOOPS bit is set to 1. When LOOPS is set, the receiver input is internally connected to the TxD pin and RSRC determines whether this connection is also connected to the transmitter output. 0 Provided LOOPS is set, RSRC is cleared, selects internal loop back mode and the SCI does not use the RxD pins. 1 Single-wire SCI mode where the TxD pin is connected to the transmitter output and receiver input.
4 M	9-Bit or 8-Bit Mode Select 0 Normal — start + 8 data bits (lsb first) + stop. 1 Receiver and transmitter use 9-bit data characters start + 8 data bits (lsb first) + 9th data bit + stop.
3 WAKE	Receiver Wakeup Method Select. Refer to Section 15.3.3.2, "Receiver Wakeup Operation" for more information. 0 Idle-line wakeup. 1 Address-mark wakeup.
2 ILT	Idle Line Type Select. Setting this bit to 1 ensures that the stop bit and logic 1 bits at the end of a character do not count toward the 10 or 11 bit times of logic high level needed by the idle line detection logic. Refer to Section 15.3.3.2.1, "Idle-Line Wakeup" for more information. 0 Idle character bit count starts after start bit. 1 Idle character bit count starts after stop bit.

Table 15-3. SCIxC1 Field Descriptions (continued)

Field	Description
1 PE	Parity Enable. Enables hardware parity generation and checking. When parity is enabled, the most significant bit (msb) of the data character (eighth or ninth data bit) is treated as the parity bit. 0 No hardware parity generation or checking. 1 Parity enabled.
0 PT	Parity Type. Provided parity is enabled (PE = 1), this bit selects even or odd parity. Odd parity means the total number of 1s in the data character, including the parity bit, is odd. Even parity means the total number of 1s in the data character, including the parity bit, is even. 0 Even parity. 1 Odd parity.

15.2.3   SCI Control Register 2 (SCIxC2)
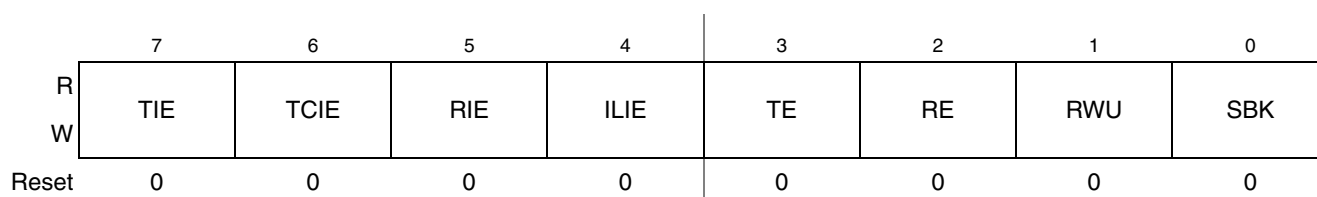
This register can be read or written at any time.

	7	6	5	4	3	2	1	0
R W	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
Reset	0	0	0	0	0	0	0	0

Figure 15-8. SCI Control Register 2 (SCIxC2)

Table 15-4. SCIxC2 Field Descriptions

Field	Description
7 TIE	Transmit Interrupt Enable (for TDRE) 0 Hardware interrupts from TDRE disabled (use polling). 1 Hardware interrupt requested when TDRE flag is 1.
6 TCIE	Transmission Complete Interrupt Enable (for TC) 0 Hardware interrupts from TC disabled (use polling). 1 Hardware interrupt requested when TC flag is 1.
5 RIE	Receiver Interrupt Enable (for RDRF) 0 Hardware interrupts from RDRF disabled (use polling). 1 Hardware interrupt requested when RDRF flag is 1.
4 ILIE	Idle Line Interrupt Enable (for IDLE) 0 Hardware interrupts from IDLE disabled (use polling). 1 Hardware interrupt requested when IDLE flag is 1.
3 TE	Transmitter Enable 0 Transmitter off. 1 Transmitter on. TE must be 1 to use the SCI transmitter. When TE is set, the SCI forces the TxD pin to act as an output for the SCI system. When the SCI is configured for single-wire operation (LOOPS = RSRC = 1), TXDIR controls the direction of traffic on the single SCI communication line (TxD pin). TE can also queue an idle character by clearing TE then setting TE while a transmission is in progress. Refer to Section 15.3.2.1, "Send Break and Queued Idle" for more details. When TE is written to 0, the transmitter keeps control of the port TxD pin until any data, queued idle, or queued break character finishes transmitting before allowing the pin to revert to a general-purpose I/O pin.

Table 15-4. SCIxC2 Field Descriptions (continued)

Field	Description
2 RE	Receiver Enable. When the SCI receiver is off, the RxD pin reverts to being a general-purpose port I/O pin. If LOOPS is set the RxD pin reverts to being a general-purpose I/O pin even if RE is set. 0 Receiver off. 1 Receiver on.
1 RWU	Receiver Wakeup Control. This bit can be written to 1 to place the SCI receiver in a standby state where it waits for automatic hardware detection of a selected wakeup condition. The wakeup condition is an idle line between messages (WAKE = 0, idle-line wakeup) or a logic 1 in the most significant data bit in a character (WAKE = 1, address-mark wakeup). Application software sets RWU and (normally) a selected hardware condition automatically clears RWU. Refer to Section 15.3.3.2, "Receiver Wakeup Operation," for more details. 0 Normal SCI receiver operation. 1 SCI receiver in standby waiting for wakeup condition.
0 SBK	Send Break. Writing a 1 and then a 0 to SBK queues a break character in the transmit data stream. Additional break characters of 10 or 11 (13 or 14 if BRK13 = 1) bit times of logic 0 are queued as long as SBK is set. Depending on the timing of the set and clear of SBK relative to the information currently being transmitted, a second break character may be queued before software clears SBK. Refer to Section 15.3.2.1, "Send Break and Queued Idle" for more details. 0 Normal transmitter operation. 1 Queue break character(s) to be sent.

15.2.4   SCI Status Register 1 (SCIxS1)

This register has eight read-only status flags. Writes have no effect. Special software sequences (which do not involve writing to this register) clear these status flags.
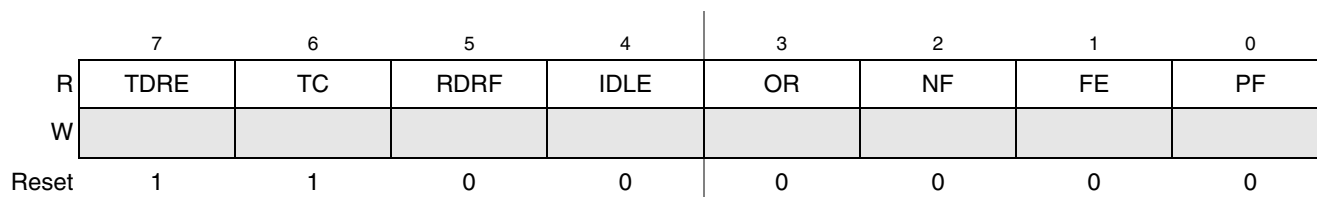
	7	6	5	4	3	2	1	0
R	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
W								
Reset	1	1	0	0	0	0	0	0

Figure 15-9. SCI Status Register 1 (SCIxS1)

Table 15-5. SCIxS1 Field Descriptions

Field	Description
7 TDRE	Transmit Data Register Empty Flag. TDRE is set out of reset and when a transmit data value transfers from the transmit data buffer to the transmit shifter, leaving room for a new character in the buffer. To clear TDRE, read SCIxS1 with TDRE set and then write to the SCI data register (SCIxD). 0 Transmit data register (buffer) full. 1 Transmit data register (buffer) empty.

6
TC
	Transmission Complete Flag. TC is set out of reset and when TDRE is set and no data, preamble, or break character is being transmitted.
	0 Transmitter active (sending data, a preamble, or a break).
	1 Transmitter idle (transmission activity complete).
	TC is cleared automatically by reading SCIxS1 with TC set and then doing one of the following:
	  • Write to the SCI data register (SCIxD) to transmit new data
	  • Queue a preamble by changing TE from 0 to 1
	  • Queue a break character by writing 1 to SBK in SCIxC2

Table 15-5. SCIxS1 Field Descriptions (continued)

Field	Description
5 RDRF	Receive Data Register Full Flag. RDRF becomes set when a character transfers from the receive shifter into the receive data register (SCIxD). To clear RDRF, read SCIxS1 with RDRF set and then read the SCI data register (SCIxD). 0 Receive data register empty. 1 Receive data register full.
4 IDLE	Idle Line Flag. IDLE is set when the SCI receive line becomes idle for a full character time after a period of activity. When ILT is cleared, the receiver starts counting idle bit times after the start bit. So if the receive character is all 1s, these bit times and the stop bit time count toward the full character time of logic high (10 or 11 bit times depending on the M control bit) needed for the receiver to detect an idle line. When ILT is set, the receiver doesn't start counting idle bit times until after the stop bit. So the stop bit and any logic high bit times at the end of the previous character do not count toward the full character time of logic high needed for the receiver to detect an idle line. To clear IDLE, read SCIxS1 with IDLE set and then read the SCI data register (SCIxD). After IDLE has been cleared, it cannot become set again until after a new character has been received and RDRF has been set. IDLE is set only once even if the receive line remains idle for an extended period. 0 No idle line detected. 1 Idle line was detected.
3 OR	Receiver Overrun Flag. OR is set when a new serial character is ready to be transferred to the receive data register (buffer), but the previously received character has not been read from SCIxD yet. In this case, the new character (and all associated error information) is lost because there is no room to move it into SCIxD. To clear OR, read SCIxS1 with OR set and then read the SCI data register (SCIxD). 0 No overrun. 1 Receive overrun (new SCI data lost).
2 NF	Noise Flag. The advanced sampling technique used in the receiver takes seven samples during the start bit and three samples in each data bit and the stop bit. If any of these samples disagrees with the rest of the samples within any bit time in the frame, the flag NF is set at the same time as RDRF is set for the character. To clear NF, read SCIxS1 and then read the SCI data register (SCIxD). 0 No noise detected. 1 Noise detected in the received character in SCIxD.
1 FE	Framing Error Flag. FE is set at the same time as RDRF when the receiver detects a logic 0 where the stop bit was expected. This suggests the receiver was not properly aligned to a character frame. To clear FE, read SCIxS1 with FE set and then read the SCI data register (SCIxD). 0 No framing error detected. This does not guarantee the framing is correct. 1 Framing error.
0 PF	Parity Error Flag. PF is set at the same time as RDRF when parity is enabled (PE = 1) and the parity bit in the received character does not agree with the expected parity value. To clear PF, read SCIxS1 and then read the SCI data register (SCIxD). 0 No parity error. 1 Parity error.

15.2.5 SCI Status Register 2 (SCIxS2)
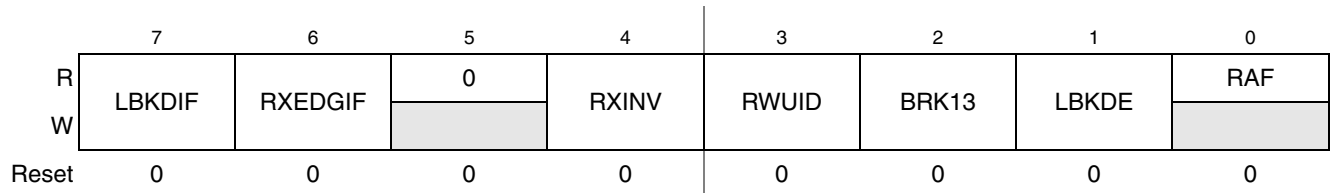
This register contains one read-only status flag.

	7	6	5	4	3	2	1	0
R	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
W								
Reset	0	0	0	0	0	0	0	0

Figure 15-10. SCI Status Register 2 (SCIxS2)

Table 15-6. SCIxS2 Field Descriptions

Field	Description
7 LBKDIF	LIN Break Detect Interrupt Flag. LBKDIF is set when the LIN break detect circuitry is enabled and a LIN break character is detected. LBKDIF is cleared by writing a 1 to it. 0 No LIN break character has been detected. 1 LIN break character has been detected.
6 RXEDGIF	RxD Pin Active Edge Interrupt Flag. RXEDGIF is set when an active edge (falling if RXINV = 0, rising if RXINV=1) on the RxD pin occurs. RXEDGIF is cleared by writing a 1 to it. 0 No active edge on the receive pin has occurred. 1 An active edge on the receive pin has occurred.
4 RXINV[1]	Receive Data Inversion. Setting this bit reverses the polarity of the received data input. 0 Receive data not inverted 1 Receive data inverted
3 RWUID	Receive Wake Up Idle Detect. RWUID controls whether the idle character that wakes up the receiver sets the IDLE bit. 0 During receive standby state (RWU = 1), the IDLE bit does not get set upon detection of an idle character. 1 During receive standby state (RWU = 1), the IDLE bit gets set upon detection of an idle character.
2 BRK13	Break Character Generation Length. BRK13 selects a longer transmitted break character length. Detection of a framing error is not affected by the state of this bit. 0 Break character is transmitted with length of 10 bit times (11 if M = 1) 1 Break character is transmitted with length of 13 bit times (14 if M = 1)
1 LBKDE	LIN Break Detection Enable. LBKDE selects a longer break character detection length. While LBKDE is set, framing error (FE) and receive data register full (RDRF) flags are prevented from setting. 0 Break character is detected at length of 10 bit times (11 if M = 1). 1 Break character is detected at length of 11 bit times (12 if M = 1).
0 RAF	Receiver Active Flag. RAF is set when the SCI receiver detects the beginning of a valid start bit, and RAF is cleared automatically when the receiver detects an idle line. This status flag can be used to check whether an SCI character is being received before instructing the MCU to go to stop mode. 0 SCI receiver idle waiting for a start bit. 1 SCI receiver active (RxD input not idle).

[1] Setting RXINV inverts the RxD input for all cases: data bits, start and stop bits, break, and idle.

When using an internal oscillator in a LIN system, it is necessary to raise the break detection threshold one bit time. Under the worst case timing conditions allowed in LIN, it is possible that a 0x00 data character can appear to be 10.26 bit times long at a slave running 14% faster than the master. This would trigger normal break detection circuitry designed to detect a 10-bit break symbol. When the LBKDE bit is set,

MCF51QE128 MCU Series Reference Manual, Rev. 3

framing errors are inhibited and the break detection threshold changes from 10 bits to 11 bits, preventing false detection of a 0x00 data character as a LIN break symbol.
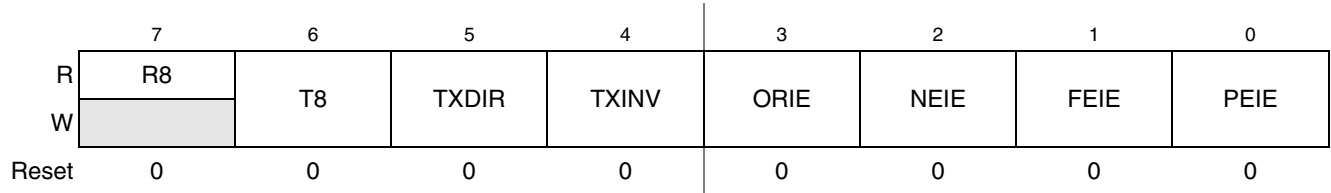
15.2.6   SCI Control Register 3 (SCIxC3)

	7	6	5	4	3	2	1	0
R	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
W								
Reset	0	0	0	0	0	0	0	0

Figure 15-11. SCI Control Register 3 (SCIxC3)

Table 15-7. SCIxC3 Field Descriptions

Field	Description
7 R8	Ninth Data Bit for Receiver. When the SCI is configured for 9-bit data (M = 1), R8 can be thought of as a ninth receive data bit to the left of the msb of the buffered data in the SCIxD register. When reading 9-bit data, read R8 before reading SCIxD because reading SCIxD completes automatic flag clearing sequences which could allow R8 and SCIxD to be overwritten with new data.
6 T8	Ninth Data Bit for Transmitter. When the SCI is configured for 9-bit data (M = 1), T8 may be thought of as a ninth transmit data bit to the left of the msb of the data in the SCIxD register. When writing 9-bit data, the entire 9-bit value is transferred to the SCI shift register after SCIxD is written so T8 should be written (if it needs to change from its previous value) before SCIxD is written. If T8 does not need to change in the new value (such as when it is used to generate mark or space parity), it need not be written each time SCIxD is written.
5 TXDIR	TxD Pin Direction in Single-Wire Mode. When the SCI is configured for single-wire half-duplex operation (LOOPS = RSRC = 1), this bit determines the direction of data at the TxD pin. 0 TxD pin is an input in single-wire mode. 1 TxD pin is an output in single-wire mode.
4 TXINV[1]	Transmit Data Inversion. Setting this bit reverses the polarity of the transmitted data output. 0 Transmit data not inverted 1 Transmit data inverted
3 ORIE	Overrun Interrupt Enable. This bit enables the overrun flag (OR) to generate hardware interrupt requests. 0 OR interrupts disabled (use polling). 1 Hardware interrupt requested when OR is set.
2 NEIE	Noise Error Interrupt Enable. This bit enables the noise flag (NF) to generate hardware interrupt requests. 0 NF interrupts disabled (use polling). 1 Hardware interrupt requested when NF is set.
1 FEIE	Framing Error Interrupt Enable. This bit enables the framing error flag (FE) to generate hardware interrupt requests. 0 FE interrupts disabled (use polling). 1 Hardware interrupt requested when FE is set.
0 PEIE	Parity Error Interrupt Enable. This bit enables the parity error flag (PF) to generate hardware interrupt requests. 0 PF interrupts disabled (use polling). 1 Hardware interrupt requested when PF is set.

[1]   Setting TXINV inverts the TxD output for all cases: data bits, start and stop bits, break, and idle.

15.2.7 SCI Data Register (SCIxD)

This register is actually two separate registers. Reads return the contents of the read-only receive data buffer and writes go to the write-only transmit data buffer. Reads and writes of this register are also involved in the automatic flag clearing mechanisms for the SCI status flags.

	7	6	5	4	3	2	1	0
R	R7	R6	R5	R4	R3	R2	R1	R0
W	T7	T6	T5	T4	T3	T2	T1	T0
Reset	0	0	0	0	0	0	0	0

Figure 15-12. SCI Data Register (SCIxD)

15.3 Functional Description

The SCI allows full-duplex, asynchronous, NRZ serial communication among the MCU and remote devices, including other MCUs. The SCI comprises a baud rate generator, transmitter, and receiver block. The transmitter and receiver operate independently, although they use the same baud rate generator. During normal operation, the MCU monitors the status of the SCI, writes the data to be transmitted, and processes received data. The following describes each of the blocks of the SCI.

15.3.1 Baud Rate Generation

As shown in Figure 15-13, the clock source for the SCI baud rate generator is the bus-rate clock.

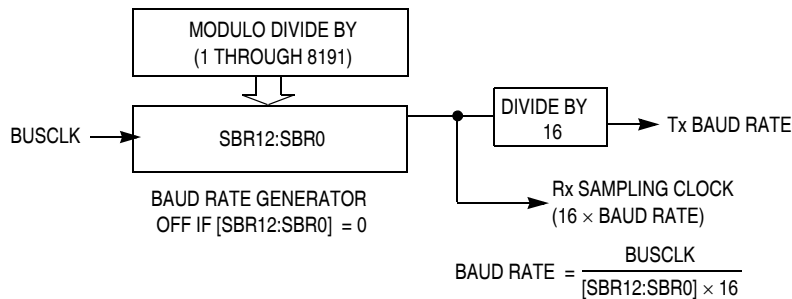$$\text{BAUD RATE} = \frac{\text{BUSCLK}}{[\text{SBR12:SBR0}] \times 16}$$

Figure 15-13. SCI Baud Rate Generation

SCI communications require the transmitter and receiver (which typically derive baud rates from independent clock sources) to use the same baud rate. Allowed tolerance on this baud frequency depends on the details of how the receiver synchronizes to the leading edge of the start bit and how bit sampling is performed.

The MCU resynchronizes to bit boundaries on every high-to-low transition. In the worst case, there are no such transitions in the full 10- or 11-bit time character frame so any mismatch in baud rate is accumulated for the whole character time. For a Freescale Semiconductor SCI system whose bus frequency is driven by a crystal, the allowed baud rate mismatch is about ±4.5 percent for 8-bit data format and about ±4 percent for 9-bit data format. Although baud rate modulo divider settings do not always produce baud rates that exactly match standard rates, it is normally possible to get within a few percent, which is acceptable for reliable communications.

15.3.2 Transmitter Functional Description

This section describes the overall block diagram for the SCI transmitter, as well as specialized functions for sending break and idle characters. The transmitter block diagram is shown in Figure 15-3.

The transmitter output (TxD) idle state defaults to logic high (TXINV is cleared following reset). The transmitter output is inverted by setting TXINV. The transmitter is enabled by setting the TE bit in SCIxC2. This queues a preamble character that is one full character frame of the idle state. The transmitter then remains idle until data is available in the transmit data buffer. Programs store data into the transmit data buffer by writing to the SCI data register (SCIxD).

The central element of the SCI transmitter is the transmit shift register that is 10 or 11 bits long depending on the setting in the M control bit. For the remainder of this section, assume M is cleared, selecting the normal 8-bit data mode. In 8-bit data mode, the shift register holds a start bit, eight data bits, and a stop bit. When the transmit shift register is available for a new SCI character, the value waiting in the transmit data register is transferred to the shift register (synchronized with the baud rate clock) and the transmit data register empty (TDRE) status flag is set to indicate another character may be written to the transmit data buffer at SCIxD.

If no new character is waiting in the transmit data buffer after a stop bit is shifted out the TxD pin, the transmitter sets the transmit complete flag and enters an idle mode, with TxD high, waiting for more characters to transmit.

Writing 0 to TE does not immediately release the pin to be a general-purpose I/O pin. Any transmit activity in progress must first be completed. This includes data characters in progress, queued idle characters, and queued break characters.

15.3.2.1 Send Break and Queued Idle

The SBK control bit in SCIxC2 sends break characters originally used to gain the attention of old teletype receivers. Break characters are a full character time of logic 0 (10 bit times including the start and stop bits). A longer break of 13 bit times can be enabled by setting BRK13. Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 1 and then write 0 to the SBK bit. This action queues a break character to be sent as soon as the shifter is available. If SBK remains 1 when the queued break moves into the shifter (synchronized to the baud rate clock), an additional break character is queued. If the receiving device is another Freescale Semiconductor SCI, the break characters are received as 0s in all eight data bits and a framing error (FE = 1) occurs.

When idle-line wakeup is used, a full character time of idle (logic 1) is needed between messages to wake up any sleeping receivers. Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 0 and then write 1 to the TE bit. This action queues an idle character to be sent as soon as the shifter is available. As long as the character in the shifter does not finish while TE is cleared, the SCI transmitter never actually releases control of the TxD pin. If there is a possibility of the shifter finishing while TE is cleard, set the general-purpose I/O controls so the pin that is shared with TxD is an output driving a logic 1. This ensures that the TxD line looks like a normal idle line even if the SCI loses control of the port pin between writing 0 and then 1 to TE.

The length of the break character is affected by the BRK13 and M bits as shown below.

Table 15-8. Break Character Length

BRK13	M	Break Character Length
0	0	10 bit times
0	1	11 bit times
1	0	13 bit times
1	1	14 bit times

15.3.3   Receiver Functional Description

In this section, the receiver block diagram (Figure 15-4) is a guide for the overall receiver functional description. Next, the data sampling technique used to reconstruct receiver data is described in more detail. Finally, two variations of the receiver wakeup function are explained.

The receiver input is inverted by setting RXINV. The receiver is enabled by setting the RE bit in SCIxC2. Character frames consist of a start bit of logic 0, eight (or nine) data bits (lsb first), and a stop bit of logic 1. For information about 9-bit data mode, refer to Section 15.3.5.1, "8- and 9-Bit Data Modes". For the remainder of this discussion, assume the SCI is configured for normal 8-bit data mode.

After receiving the stop bit into the receive shifter, and provided the receive data register is not already full, the data character is transferred to the receive data register and the receive data register full (RDRF) status flag is set. If RDRF was already set indicating the receive data register (buffer) was already full, the overrun (OR) status flag is set and the new data is lost. Because the SCI receiver is double-buffered, the program has one full character time after RDRF is set before the data in the receive data buffer must be read to avoid a receiver overrun.

When a program detects that the receive data register is full (RDRF = 1), it gets the data from the receive data register by reading SCIxD. The RDRF flag is cleared automatically by a two-step sequence normally satisfied in the course of the user's program that manages receive data. Refer to Section 15.3.4, "Interrupts and Status Flags," for more details about flag clearing.

15.3.3.1   Data Sampling Technique

The SCI receiver uses a 16× baud rate clock for sampling. The receiver starts by taking logic level samples at 16 times the baud rate to search for a falling edge on the RxD serial data input pin. A falling edge is defined as a logic 0 sample after three consecutive logic 1 samples. The 16× baud rate clock divides the bit time into 16 segments labeled RT1 through RT16. When a falling edge is located, three more samples are taken at RT3, RT5, and RT7 to make sure this was a real start bit and not merely noise. If at least two of these three samples are 0, the receiver assumes it is synchronized to a receive character.

The receiver then samples each bit time, including the start and stop bits, at RT8, RT9, and RT10 to determine the logic level for that bit. The logic level is interpreted to be that of the majority of the samples taken during the bit time. In the case of the start bit, the bit is assumed to be 0 if at least two of the samples at RT3, RT5, and RT7 are 0 even if one or all of the samples taken at RT8, RT9, and RT10 are 1s. If any sample in any bit time (including the start and stop bits) in a character frame fails to agree with the logic

level for that bit, the noise flag (NF) is set when the received character is transferred to the receive data buffer.

The falling edge detection logic continuously looks for falling edges. If an edge is detected, the sample clock is resynchronized to bit times. This improves the reliability of the receiver in the presence of noise or mismatched baud rates. It does not improve worst case analysis because some characters do not have any extra falling edges anywhere in the character frame.

In the case of a framing error, provided the received character was not a break character, the sampling logic that searches for a falling edge is filled with three logic 1 samples so that a new start bit can be detected almost immediately.

In the case of a framing error, the receiver is inhibited from receiving any new characters until the framing error flag is cleared. The receive shift register continues to function, but a complete character cannot transfer to the receive data buffer if FE is still set.

15.3.3.2   Receiver Wakeup Operation

Receiver wakeup is a hardware mechanism that allows an SCI receiver to ignore the characters in a message that is intended for a different SCI receiver. In such a system, all receivers evaluate the first character(s) of each message, and as soon as they determine the message is intended for a different receiver, they write logic 1 to the receiver wake up (RWU) control bit in SCIxC2. When RWU bit is set, the status flags associated with the receiver (with the exception of the idle bit, IDLE, when RWUID bit is set) are inhibited from setting, thus eliminating the software overhead for handling the unimportant message characters. At the end of a message, or at the beginning of the next message, all receivers automatically force RWU to 0 so all receivers wake up in time to look at the first character(s) of the next message.

15.3.3.2.1   Idle-Line Wakeup

When wake is cleared, the receiver is configured for idle-line wakeup. In this mode, RWU is cleared automatically when the receiver detects a full character time of the idle-line level. The M control bit selects 8-bit or 9-bit data mode that determines how many bit times of idle are needed to constitute a full character time (10 or 11 bit times because of the start and stop bits).

When RWU is one and RWUID is zero, the idle condition that wakes up the receiver does not set the IDLE flag. The receiver wakes up and waits for the first data character of the next message that sets the RDRF flag and generates an interrupt if enabled. When RWUID is one, any idle condition sets the IDLE flag and generates an interrupt if enabled, regardless of whether RWU is zero or one.

The idle-line type (ILT) control bit selects one of two ways to detect an idle line. When ILT is cleared, the idle bit counter starts after the start bit so the stop bit and any logic 1s at the end of a character count toward the full character time of idle. When ILT is set, the idle bit counter does not start until after a stop bit time, so the idle detection is not affected by the data in the last character of the previous message.

15.3.3.2.2   Address-Mark Wakeup

When wake is set, the receiver is configured for address-mark wakeup. In this mode, RWU is cleared automatically when the receiver detects a logic 1 in the most significant bit of a received character (eighth bit when M is cleared and ninth bit when M is set).

Address-mark wakeup allows messages to contain idle characters, but requires the msb be reserved for use in address frames. The logic 1 msb of an address frame clears the RWU bit before the stop bit is received and sets the RDRF flag. In this case, the character with the msb set is received even though the receiver was sleeping during most of this character time.

15.3.4   Interrupts and Status Flags

The SCI system has three separate interrupt vectors to reduce the amount of software needed to isolate the cause of the interrupt. One interrupt vector is associated with the transmitter for TDRE and TC events. Another interrupt vector is associated with the receiver for RDRF, IDLE, RXEDGIF, and LBKDIF events. A third vector is used for OR, NF, FE, and PF error conditions. Each of these ten interrupt sources can be separately masked by local interrupt enable masks. The flags can be polled by software when the local masks are cleared to disable generation of hardware interrupt requests.

The SCI transmitter has two status flags that can optionally generate hardware interrupt requests. Transmit data register empty (TDRE) indicates when there is room in the transmit data buffer to write another transmit character to SCIxD. If the transmit interrupt enable (TIE) bit is set, a hardware interrupt is requested when TDRE is set. Transmit complete (TC) indicates that the transmitter is finished transmitting all data, preamble, and break characters and is idle with TxD at the inactive level. This flag is often used in systems with modems to determine when it is safe to turn off the modem. If the transmit complete interrupt enable (TCIE) bit is set, a hardware interrupt is requested when TC is set. Instead of hardware interrupts, software polling may be used to monitor the TDRE and TC status flags if the corresponding TIE or TCIE local interrupt masks are cleared.

When a program detects that the receive data register is full (RDRF = 1), it gets the data from the receive data register by reading SCIxD. The RDRF flag is cleared by reading SCIxS1 while RDRF is set and then reading SCIxD.

When polling is used, this sequence is naturally satisfied in the normal course of the user program. If hardware interrupts are used, SCIxS1 must be read in the interrupt service routine (ISR). Normally, this is done in the ISR anyway to check for receive errors, so the sequence is automatically satisfied.

The IDLE status flag includes logic that prevents it from getting set repeatedly when the RxD line remains idle for an extended period of time. IDLE is cleared by reading SCIxS1 while IDLE is set and then reading SCIxD. After IDLE has been cleared, it cannot become set again until the receiver has received at least one new character and has set RDRF.

If the associated error was detected in the received character that caused RDRF to be set, the error flags — noise flag (NF), framing error (FE), and parity error flag (PF) — are set at the same time as RDRF. These flags are not set in overrun cases.

If RDRF was already set when a new character is ready to be transferred from the receive shifter to the receive data buffer, the overrun (OR) flag is set instead of the data along with any associated NF, FE, or PF condition is lost.

At any time, an active edge on the RxD serial data input pin causes the RXEDGIF flag to set. The RXEDGIF flag is cleared by writing a 1 to it. This function does depend on the receiver being enabled (RE = 1).

15.3.5 Additional SCI Functions

The following sections describe additional SCI functions.

15.3.5.1 8- and 9-Bit Data Modes

The SCI system (transmitter and receiver) can be configured to operate in 9-bit data mode by setting the M control bit in SCIxC1. In 9-bit mode, there is a ninth data bit to the left of the msb of the SCI data register. For the transmit data buffer, this bit is stored in T8 in SCIxC3. For the receiver, the ninth bit is held in R8 in SCIxC3.

For coherent writes to the transmit data buffer, write to the T8 bit before writing to SCIxD.

If the bit value to be transmitted as the ninth bit of a new character is the same as for the previous character, it is not necessary to write to T8 again. When data is transferred from the transmit data buffer to the transmit shifter, the value in T8 is copied at the same time data is transferred from SCIxD to the shifter.

The 9-bit data mode is typically used with parity to allow eight bits of data plus the parity in the ninth bit, or it is used with address-mark wakeup so the ninth data bit can serve as the wakeup bit. In custom protocols, the ninth bit can also serve as a software-controlled marker.

15.3.5.2 Stop Mode Operation

During all stop modes, clocks to the SCI module are halted.

In stop1 and stop2 modes, all SCI register data is lost and must be re-initialized upon recovery from these two stop modes. No SCI module registers are affected in stop3 mode.

The receive input active edge detect circuit remains active in stop3 mode, but not in stop2. An active edge on the receive input brings the CPU out of stop3 mode if the interrupt is not masked (RXEDGIE = 1).

Because the clocks are halted, the SCI module resumes operation upon exit from stop (only in stop3 mode). Software should ensure stop mode is not entered while there is a character being transmitted out of or received into the SCI module.

15.3.5.3 Loop Mode

When LOOPS is set, the RSRC bit in the same register chooses between loop mode (RSRC = 0) or single-wire mode (RSRC = 1). Loop mode is sometimes used to check software, independent of connections in the external system, to help isolate system problems. In this mode, the transmitter output is

internally connected to the receiver input and the RxD pin is not used by the SCI, so it reverts to a general-purpose port I/O pin.

15.3.5.4   Single-Wire Operation

When LOOPS is set, the RSRC bit in the same register chooses between loop mode (RSRC = 0) or single-wire mode (RSRC = 1). Single-wire mode implements a half-duplex serial connection. The receiver is internally connected to the transmitter output and to the TxD pin. The RxD pin is not used and reverts to a general-purpose port I/O pin.

In single-wire mode, the TXDIR bit in SCIxC3 controls the direction of serial data on the TxD pin. When TXDIR is cleared, the TxD pin is an input to the SCI receiver and the transmitter is temporarily disconnected from the TxD pin so an external device can send serial data to the receiver. When TXDIR is set, the TxD pin is an output driven by the transmitter. In single-wire mode, the internal loop back connection from the transmitter to the receiver causes the receiver to receive characters that are sent out by the transmitter.

Chapter 16
Serial Peripheral Interface (S08SPIV3)

16.1 Introduction

Figure 16-1 shows the MCF51QE128 Series block diagram with the SPI highlighted.

NOTE

Ignore any references to stop1 low-power mode in this chapter, because the MCF51QE128 device does not support it.

16.1.1 SPI Clock Gating

The bus clock to SPI1 and SPI2 can be gated on and off using the SPI1 and SPI2 bits, respectively, in SCGC2. These bits are set after any reset, which enables the bus clock to this module. To conserve power, these bits can be cleared to disable the clock to either of these modus when not in use. See Section 5.6, "Peripheral Clock Gating," for details.

16.1.2 Interrupt Vector

See Chapter 8, "Interrupt Controller (CF1_INTC)," for the SPI interrupt vector assignments.
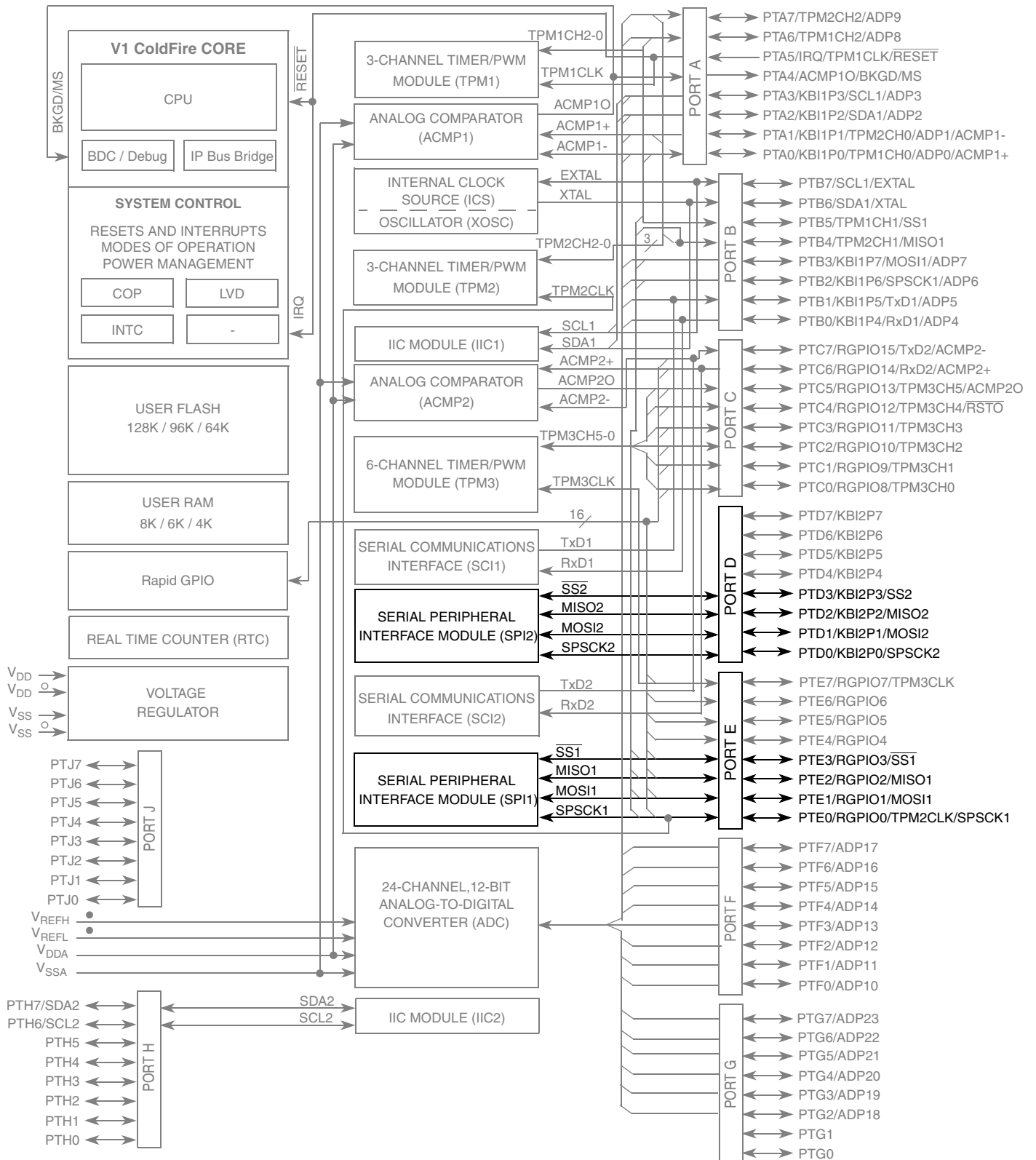
Figure 16-1. MCF51QE128 Block Diagram Highlighting SPI Block and Pins

16.1.3   Features

Features of the SPI module include:

- Master or slave mode operation
- Full-duplex or single-wire bidirectional option
- Programmable transmit bit rate
- Double-buffered transmit and receive
- Serial clock phase and polarity options
- Slave select output
- Selectable msb-first or lsb-first shifting

16.1.4   Block Diagrams

This section includes block diagrams showing SPI system connections, the internal organization of the SPI module, and the SPI clock dividers that control the master mode bit rate.

16.1.4.1   SPI System Block Diagram

Figure 16-2 shows the SPI modules of two MCUs connected in a master-slave arrangement. The master device initiates all SPI data transfers. During a transfer, the master shifts data out (on the MOSI pin) to the slave while simultaneously shifting data in (on the MISO pin) from the slave. The transfer effectively exchanges the data that was in the SPI shift registers of the two SPI systems. The SPSCK signal is a clock output from the master and an input to the slave. The slave device must be selected by a low level on the slave select input ($\overline{SS}$ pin). In this system, the master device has configured its $\overline{SS}$ pin as an optional slave select output.
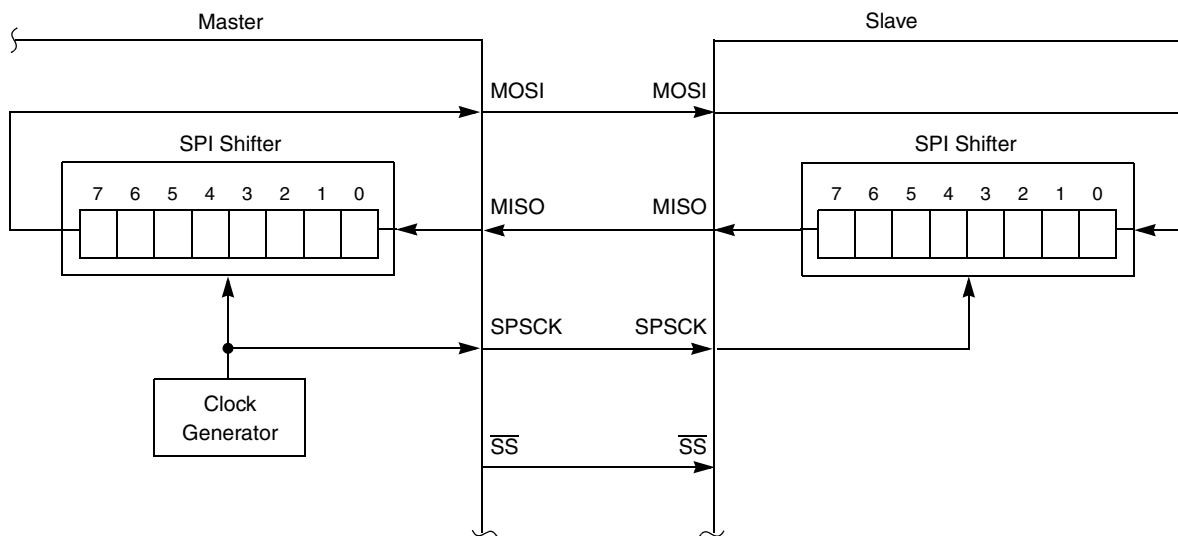

Figure 16-2. SPI System Connections

The most common uses of the SPI system include connecting simple shift registers for adding input or output ports or connecting small peripheral devices such as serial A/D or D/A converters. Although

Figure 16-2 shows a system where data is exchanged between two MCUs, many practical systems involve simpler connections where data is unidirectionally transferred from the master MCU to a slave or from a slave to the master MCU.

16.1.4.2   SPI Module Block Diagram

Figure 16-3 is a block diagram of the SPI module. The central element of the SPI is the SPI shift register. Data is written to the double-buffered transmitter (write to SPIxD) and gets transferred to the SPI shift register at the start of a data transfer. After shifting in a byte of data, the data is transferred into the double-buffered receiver where it can be read (read from SPIxD). Pin multiplexing logic controls connections between MCU pins and the SPI module.

When the SPI is configured as a master, the clock output is routed to the SPSCK pin, the shifter output is routed to MOSI, and the shifter input is routed from the MISO pin.

When the SPI is configured as a slave, the SPSCK pin is routed to the clock input of the SPI, the shifter output is routed to MISO, and the shifter input is routed from the MOSI pin.

In the external SPI system, simply connect all SPSCK pins to each other, all MISO pins together, and all MOSI pins together. Peripheral devices often use slightly different names for these pins.

Figure 16-3. SPI Module Block Diagram

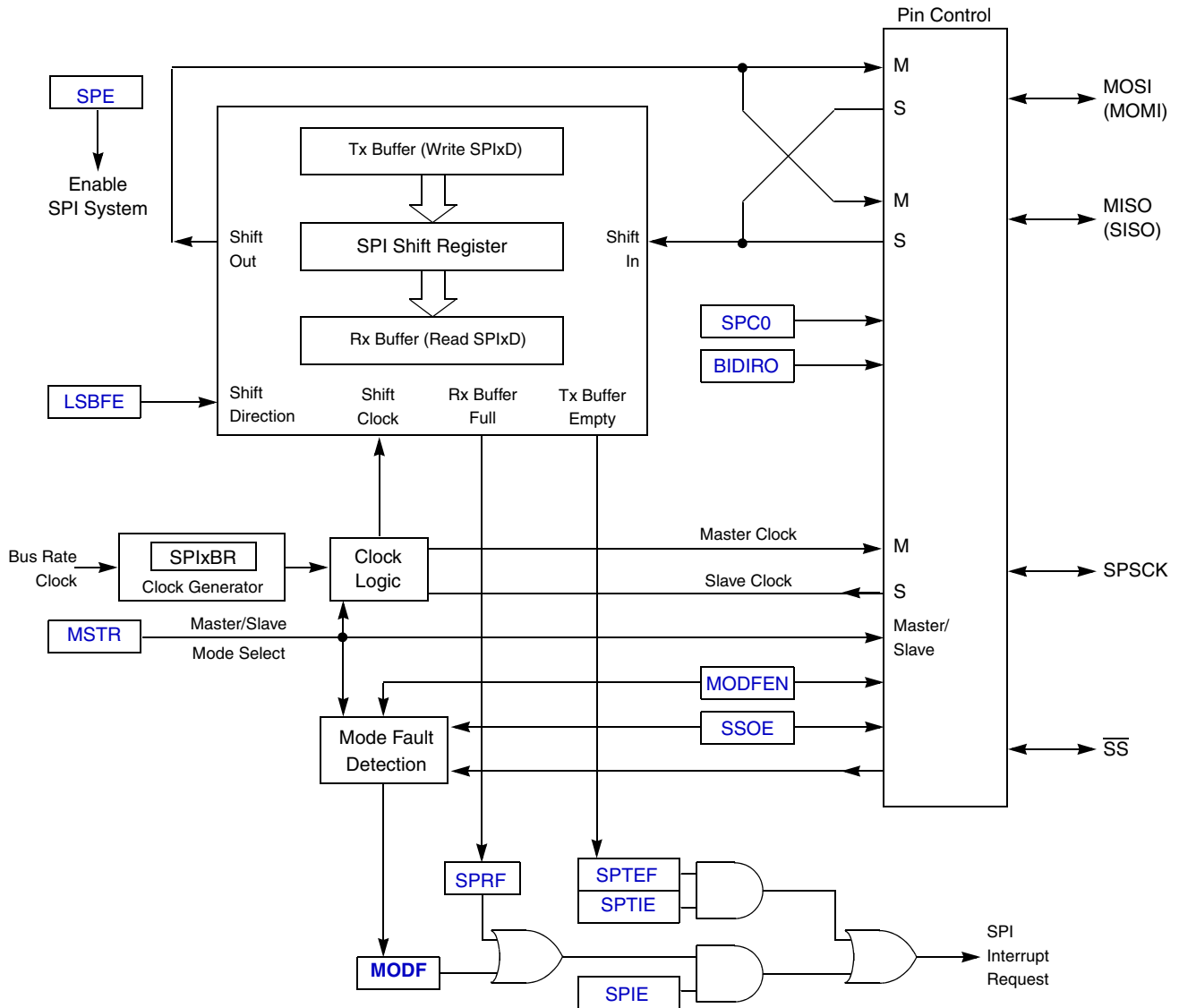## 16.1.5   SPI Baud Rate Generation

As shown in Figure 16-4, the clock source for the SPI baud rate generator is the bus clock. The three prescale bits (SPIxBR[SPPR]) choose a prescale divisor of 1, 2, 3, 4, 5, 6, 7, or 8. The three rate select bits (SPIxBR[SPR]) divide the output of the prescaler stage by 2, 4, 8, 16, 32, 64, 128, or 256 to obtain the internal SPI master mode bit-rate clock.
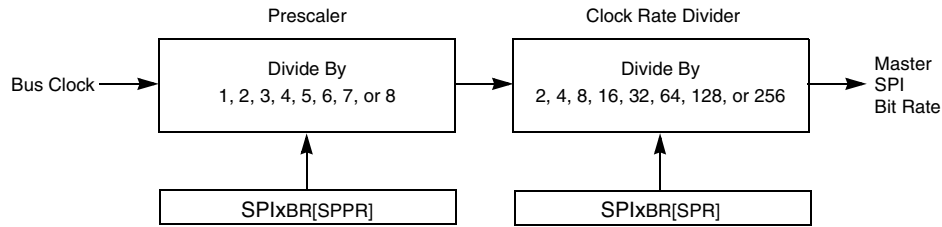
Figure 16-4. SPI Baud Rate Generation

16.2 External Signal Description

The SPI optionally shares four port pins. The function of these pins depends on the settings of SPI control bits. When the SPI is disabled (SPIxC1[SPE] is cleared), these four pins revert to being general-purpose port I/O pins that are not controlled by the SPI.

16.2.1 SPSCK — SPI Serial Clock

When the SPI is enabled as a slave, this pin is the serial clock input. When the SPI is enabled as a master, this pin is the serial clock output.

16.2.2 MOSI — Master Data Out, Slave Data In

When the SPI is enabled as a master and SPI pin control zero bit (SPIxC2[SPC0]) is cleared (not bidirectional mode), this pin is the serial data output. When the SPI is enabled as a slave and SPIxC2[SPC0] is cleared, this pin is the serial data input. If SPC0 is set to select single-wire bidirectional mode and master mode is selected, this pin becomes the bidirectional data I/O pin (MOMI). Also, the bidirectional mode output enable bit determines whether the pin acts as an input (SPIxC2[BIDIROE] = 0) or an output (BIDIROE = 1). If SPC0 is set and slave mode is selected, this pin is not used by the SPI and reverts to a general-purpose port I/O pin.

16.2.3 MISO — Master Data In, Slave Data Out

When the SPI is enabled as a master and SPI pin control zero (SPIxC2[SPC0]) is cleared (not bidirectional mode), this pin is the serial data input. When the SPI is enabled as a slave and SPC0 is cleared, this pin is the serial data output. If SPC0 is set to select single-wire bidirectional mode and slave mode is selected, this pin becomes the bidirectional data I/O pin (SISO) and the bidirectional mode output enable bit determines whether the pin acts as an input (BIDIROE = 0) or an output (BIDIROE = 1). If SPC0 is set and master mode is selected, this pin is not used by the SPI and reverts to being a general-purpose port I/O pin.

16.2.4 $\overline{SS}$ — Slave Select

When the SPI is enabled as a slave, this pin is the low-true slave select input. When the SPI is enabled as a master and mode fault enable is off (SPIxC2[MODFEN] = 0), this pin is not used by the SPI and reverts to a general-purpose port I/O pin. When the SPI is enabled as a master and MODFEN is set, the slave select

output enable bit determines whether this pin acts as the mode fault input (SPIxC1[SSOE] = 0) or as the slave select output (SSOE = 1).

16.3   Modes of Operation

16.3.1   SPI in Stop Modes

The SPI is disabled in all stop modes, regardless of the settings before executing the STOP instruction. During stop2 mode, the SPI module is fully powered down. Upon wake-up from stop2 mode, the SPI module is in the reset state. During stop3 mode, clocks to the SPI module are halted. No registers are affected. If stop3 is exited with a reset, the SPI is placed into its reset state. If stop3 is exited with an interrupt, the SPI continues from the state it was in when stop3 was entered.

16.4   Register Definition

The SPI contains five 8-bit registers to select SPI options, control baud rate, report SPI status, and for transmit/receive data.

Refer to the direct-page register summary in the memory chapter of this document for the absolute address assignments for all SPI registers. This section refers to registers and control bits only by their names, and a Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

16.4.1   SPI Control Register 1 (SPIxC1)

This read/write register includes the SPI enable control, interrupt enables, and configuration options.

	7	6	5	4	3	2	1	0
R W	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
Reset	0	0	0	0	0	1	0	0
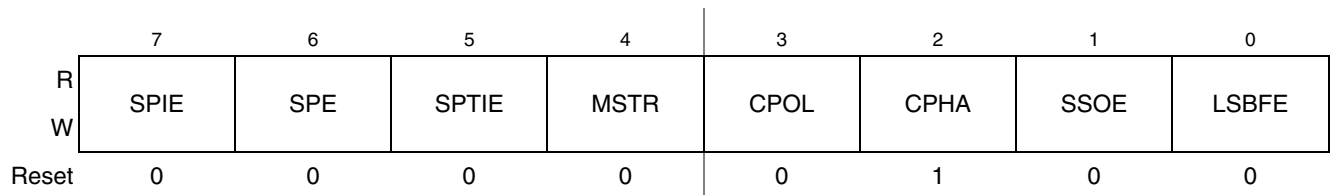
Figure 16-5. SPI Control Register 1 (SPIxC1)

Table 16-1. SPIxC1 Field Descriptions

Field	Description
7 SPIE	SPI Interrupt Enable (for SPRF and MODF). This is the interrupt enable for SPI receive buffer full (SPRF) and mode fault (MODF) events. 0 Interrupts from SPRF and MODF inhibited (use polling) 1 When SPRF or MODF is 1, request a hardware interrupt
6 SPE	SPI System Enable. Disabling the SPI halts any transfer in progress, clears data buffers, and initializes internal state machines. SPRF is cleared and SPTEF is set to indicate the SPI transmit data buffer is empty. 0 SPI system inactive 1 SPI system enabled

Table 16-1. SPIxC1 Field Descriptions (continued)

Field	Description
5 SPTIE	SPI Transmit Interrupt Enable. This is the interrupt enable bit for SPI transmit buffer empty (SPTEF). 0 Interrupts from SPTEF inhibited (use polling) 1 When SPTEF is 1, hardware interrupt requested
4 MSTR	Master/Slave Mode Select 0 SPI module configured as a slave SPI device 1 SPI module configured as a master SPI device
3 CPOL	Clock Polarity. This bit effectively places an inverter in series with the clock signal from a master SPI or to a slave SPI device. Refer to Section 16.5.1, "SPI Clock Formats" for more details. 0 Active-high SPI clock (idles low) 1 Active-low SPI clock (idles high)
2 CPHA	Clock Phase. This bit selects one of two clock formats for different synchronous serial peripheral devices. Refer to Section 16.5.1, "SPI Clock Formats" for more details. 0 First edge on SPSCK occurs at the middle of the first cycle of an 8-cycle data transfer 1 First edge on SPSCK occurs at the start of the first cycle of an 8-cycle data transfer
1 SSOE	Slave Select Output Enable. This bit is used with the mode fault enable (SPIxC2[MODFEN]) bit and the master/slave (MSTR) control bit to determine the function of the SS pin as shown in Table 16-2.
0 LSBFE	lsb First (Shifter Direction) 0 SPI serial data transfers start with most significant bit 1 SPI serial data transfers start with least significant bit

Table 16-2. $\overline{SS}$ Pin Function

MODFEN	SSOE	Master Mode	Slave Mode
0	0	General-purpose I/O (not SPI)	Slave select input
0	1	General-purpose I/O (not SPI)	Slave select input
1	0	$\overline{SS}$ input for mode fault	Slave select input
1	1	Automatic $\overline{SS}$ output	Slave select input

NOTE

Ensure that the SPI should not be disabled (SPE = 0) at the same time as a bit change to SPIxC1[CPHA]. These changes should be performed as separate operations or unexpected behavior may occur.

16.4.2   SPI Control Register 2 (SPIxC2)

This read/write register controls optional features of the SPI system. Bits 7, 6, 5, and 2 are reserved and always read 0.

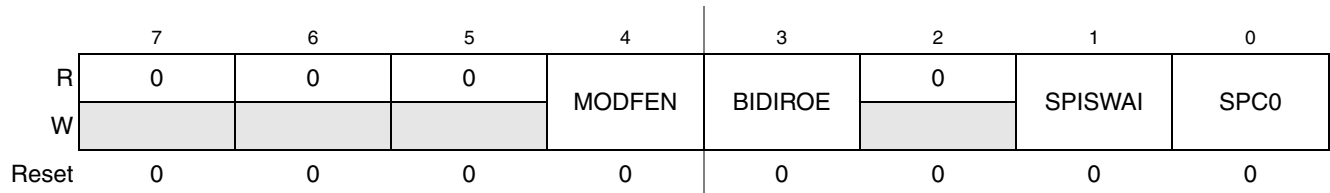|	7	6	5	4	3	2	1	0
R	0	0	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
W								
Reset	0	0	0	0	0	0	0	0

Figure 16-6. SPI Control Register 2 (SPIxC2)

Table 16-3. SPIxC2 Register Field Descriptions

Field	Description
7–5	Reserved, should be cleared.
4 MODFEN	Master Mode-Fault Function Enable. When the SPI is configured for slave mode, this bit has no meaning or effect. (The SS pin is the slave select input.) In master mode, this bit determines how the SS pin is used (refer to Table 16-2 for more details). 0 Mode fault function disabled, master SS pin reverts to general-purpose I/O not controlled by SPI 1 Mode fault function enabled, master SS pin acts as the mode fault input or the slave select output
3 BIDIROE	Bidirectional Mode Output Enable. When bidirectional mode is enabled by setting SPC0, BIDIROE determines whether the SPI data output driver is enabled to the single bidirectional SPI I/O pin. Depending on whether the SPI is configured as a master or a slave, it uses the MOSI (MOMI) or MISO (SISO) pin, respectively, as the single SPI data I/O pin. When SPC0 is cleared, BIDIROE has no meaning or effect. 0 Output driver disabled so SPI data I/O pin acts as an input 1 SPI I/O pin enabled as an output
2	Reserved, should be cleared.
1 SPISWAI	**SPI Stop in Wait Mode** 0 SPI clocks continue to operate in wait mode 1 SPI clocks stop when the MCU enters wait mode
0 SPC0	**SPI Pin Control 0** — The SPC0 bit chooses single-wire bidirectional mode. If MSTR is cleared (slave mode), the SPI uses the MISO (SISO) pin for bidirectional SPI data transfers. If MSTR is set (master mode), the SPI uses the MOSI (MOMI) pin for bidirectional SPI data transfers. When SPC0 is set 1, BIDIROE enables or disables the output driver for the single bidirectional SPI I/O pin. 0 SPI uses separate pins for data input and data output 1 SPI configured for single-wire bidirectional operation

16.4.3   SPI Baud Rate Register (SPIxBR)

This register sets the prescaler and bit rate divisor for an SPI master. This register may be read or written at any time.
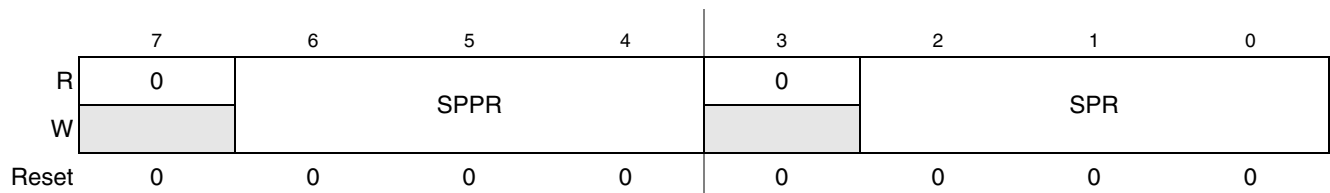
	7	6	5	4	3	2	1	0
R	0		SPPR		0		SPR	
W								
Reset	0	0	0	0	0	0	0	0

Figure 16-7. SPI Baud Rate Register (SPIxBR)

Table 16-4. SPIxBR Register Field Descriptions

Field	Description
7	Reserved, should be cleared.
6–4 SPPR	SPI Baud Rate Prescale Divisor. This 3-bit field selects one of eight divisors for the SPI baud rate prescaler as shown below. The input to this prescaler is the bus rate clock (BUSCLK). The output of this prescaler drives the input of the SPI baud rate divider (see Figure 16-4). 	SPPR	Prescaler Divisor	SPPR	Prescaler Divisor	 |------|-------------------|------|-------------------| 	000	1	100	5	 	001	2	101	6	 	010	3	110	7	 	011	4	111	8	
3	Reserved, should be cleared.
2–0 SPR	SPI Baud Rate Divisor. This 3-bit field selects one of eight divisors for the SPI baud rate divider as shown below. The SPI baud rate prescaler supplies the input to this divider (see Figure 16-4). The output of this divider is the SPI bit rate clock for master mode. 	SPR	Rate Divisor	SPR	Rate Divisor	 |-----|--------------|-----|--------------| 	000	2	100	32	 	001	4	101	64	 	010	8	110	128	 	011	16	111	256	

16.4.4   SPI Status Register (SPIxS)

This register has three read-only status bits. Bits 6, 3, 2, 1, and 0 are not reserved and always read 0. Writes have no meaning or effect.

	7	6	5	4	3	2	1	0
R	SPRF	0	SPTEF	MODF	0	0	0	0
W								
Reset	0	0	1	0	0	0	0	0

Figure 16-8. SPI Status Register (SPIxS)

Table 16-5. SPIxS Register Field Descriptions

Field	Description
7 SPRF	SPI Read Buffer Full Flag. SPRF is set at the completion of an SPI transfer to indicate that received data may be read from the SPI data register (SPIxD). SPRF is cleared by reading SPRF while it is set, then reading SPIxD. 0 No data available in the receive data buffer 1 Data available in the receive data buffer
6	Reserved, should be cleared.

Table 16-5. SPIxS Register Field Descriptions

Field	Description
5 SPTEF	SPI Transmit Buffer Empty Flag. This bit is set when there is room in the transmit data buffer. It is cleared by reading SPIxS with SPTEF set, followed by writing a data value to the transmit buffer at SPIxD. SPIxS must be read with SPTEF set before writing data to SPIxD or the SPIxD write is ignored. SPTEF generates a CPU interrupt request if SPIxC1[SPTIE] is also set. SPTEF is automatically set when a data byte transfers from the transmit buffer into the transmit shift register. For an idle SPI (no data in the transmit buffer or the shift register and no transfer in progress), data written to SPIxD is transferred to the shifter almost immediately so SPTEF is set within two bus cycles allowing a second 8-bit data value to be queued into the transmit buffer. After completion of the transfer of the value in the shift register, the queued value from the transmit buffer is automatically moved to the shifter and SPTEF is set, indicating there is room for new data in the transmit buffer. If no new data is waiting in the transmit buffer, SPTEF remains set and no data moves from the buffer to the shifter. 0 SPI transmit buffer not empty 1 SPI transmit buffer empty
4 MODF	Master Mode Fault Flag. MODF is set if the SPI is configured as a master and the slave select input asserts, indicating some other SPI device is also configured as a master. The SS pin acts as a mode fault error input only when MSTR and MODFEN are set and SSOE is cleared. Otherwise, MODF is never set. MODF is cleared by reading MODF while it is 1, then writing to the SPIxC1 register. 0 No mode fault error 1 Mode fault error detected
3–0	Reserved, should be cleared.

16.4.5 SPI Data Register (SPIxD)

Reads of this register returns the data read from the receive data buffer. Writes to this register write data to the transmit data buffer. When the SPI is configured as a master, writing data to the transmit data buffer initiates an SPI transfer.

Data should not be written to the transmit data buffer unless SPIxS[SPTEF] is set, indicating there is room in the transmit buffer to queue a new transmit byte.

Data may be read from SPIxD any time after SPRF is set and before another transfer is finished. Failure to read the data out of the receive data buffer before a new transfer ends causes a receive overrun condition and the data from the new transfer is lost.

	7	6	5	4	3	2	1	0
R								
W				Data				
Reset	0	0	0	0	0	0	0	0

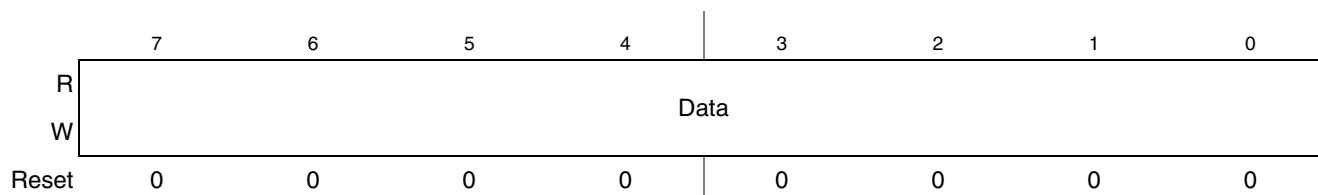**Figure 16-9. SPI Data Register (SPIxD)**

16.5 Functional Description

An SPI transfer is initiated by checking for the SPI transmit buffer empty flag (SPIxS[SPTEF] = 1) and then writing a byte of data to the SPI data register (SPIxD) in the master SPI device. When the SPI shift register is available, this byte of data is moved from the transmit data buffer to the shifter, SPTEF is set to

indicate there is room in the buffer to queue another transmit character if desired, and the SPI serial transfer starts.

During the SPI transfer, data is sampled (read) on the MISO pin at one SPSCK edge and shifted, changing the bit value on the MOSI pin, one-half SPSCK cycle later. After eight SPSCK cycles, the data that was in the shift register of the master has been shifted out the MOSI pin to the slave while eight bits of data were shifted in the MISO pin into the master's shift register. At the end of this transfer, the received data byte is moved from the shifter into the receive data buffer and SPRF is set to indicate the data can be read by reading SPIxD. If another byte of data is waiting in the transmit buffer at the end of a transfer, it is moved into the shifter, SPTEF is set, and a new transfer is started.

Normally, SPI data is transferred most significant bit (msb) first. If SPIxC1[LSBFE] is set, SPI data is shifted lsb first.

When the SPI is configured as a slave, its $\overline{SS}$ pin must be driven low before a transfer starts and $\overline{SS}$ must stay low throughout the transfer. If a clock format where CPHA is cleared is selected, $\overline{SS}$ must be driven to a logic 1 between successive transfers. If CPHA is set, $\overline{SS}$ may remain low between successive transfers. See Section 16.5.1, "SPI Clock Formats," for more details.

Because the transmitter and receiver are double buffered, a second byte, in addition to the byte currently being shifted out, can be queued into the transmit data buffer. A previously received character can be in the receive data buffer while a new character is being shifted in. The SPTEF flag indicates when the transmit buffer has room for a new character. The SPRF flag indicates when a received character is available in the receive data buffer. The received character must be read out of the receive buffer (read SPIxD) before the next transfer is finished or a receive overrun error results.

In the case of a receive overrun, the new data is lost because the receive buffer held the previous character and was not ready to accept the new data. There is no indication for such an overrun condition so the user must ensure that previous data has been read from the receive buffer before a new transfer is initiated.

16.5.1   SPI Clock Formats

To accommodate a wide variety of synchronous serial peripherals from different manufacturers, the SPI system has a clock polarity (CPOL) bit and a clock phase (CPHA) control bit to select one of four clock formats for data transfers. CPOL selectively inserts an inverter in series with the clock. CPHA chooses between two different clock phase relationships between the clock and data.

Figure 16-10 shows the clock formats when CPHA is set. At the top of the figure, the eight bit times are shown for reference with bit 1 starting at the first SPSCK edge and bit 8 ending one-half SPSCK cycle after the sixteenth SPSCK edge. The msb first and lsb first lines show the order of SPI data bits depending on SPIxC1[LSBFE]. Both variations of SPSCK polarity are shown; however, only one of these waveforms applies for a specific transfer, depending on the value in CPOL. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The $\overline{SS}$ OUT waveform applies to the slave select output from a master (provided MODFEN and SSOE are set). The master $\overline{SS}$ output asserts one-half SPSCK cycle before the start of the transfer and negates at the end of the eighth bit time of the transfer. The $\overline{SS}$ IN waveform applies to the slave select input of a slave.

Figure 16-10. SPI Clock Formats (CPHA = 1)

When CPHA is set, the slave begins to drive its MISO output when $\overline{SS}$ asserts, but the data is not defined until the first SPSCK edge. The first SPSCK edge shifts the first bit of data from the shifter onto the MOSI output of the master and the MISO output of the slave. The next SPSCK edge causes the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the third SPSCK edge, the SPI shifter shifts one bit position that shifts in the bit value that was sampled, and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When CHPA is set, the slave's $\overline{SS}$ input is not required to negate between transfers.

Figure 16-11 shows the clock formats when CPHA is cleared. At the top of the figure, the eight bit times are shown for reference with bit 1 starting as the slave is selected ($\overline{SS}$ IN asserts), and bit 8 ends at the last SPSCK edge. The msb first and lsb first lines show the order of SPI data bits depending on the setting of SPIxC1[LSBFE]. Both variations of SPSCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The $\overline{SS}$ OUT waveform applies to the slave select output from a master (provided MODFEN and SSOE are set). The master $\overline{SS}$ output asserts at the start of the first bit time of the transfer and negates one-half SPSCK cycle after the end of the eighth bit time of the transfer. The $\overline{SS}$ IN waveform applies to the slave select input of a slave.

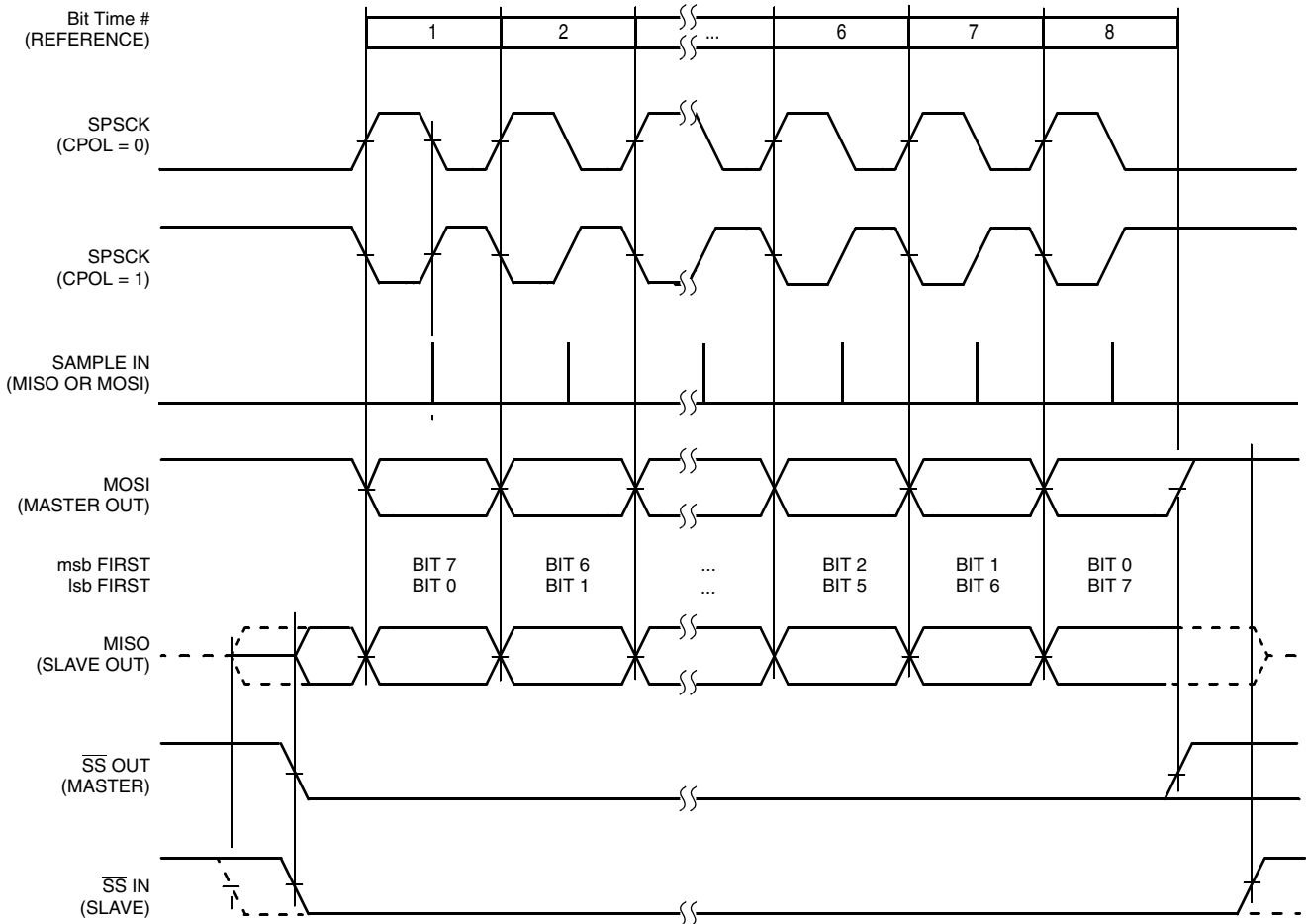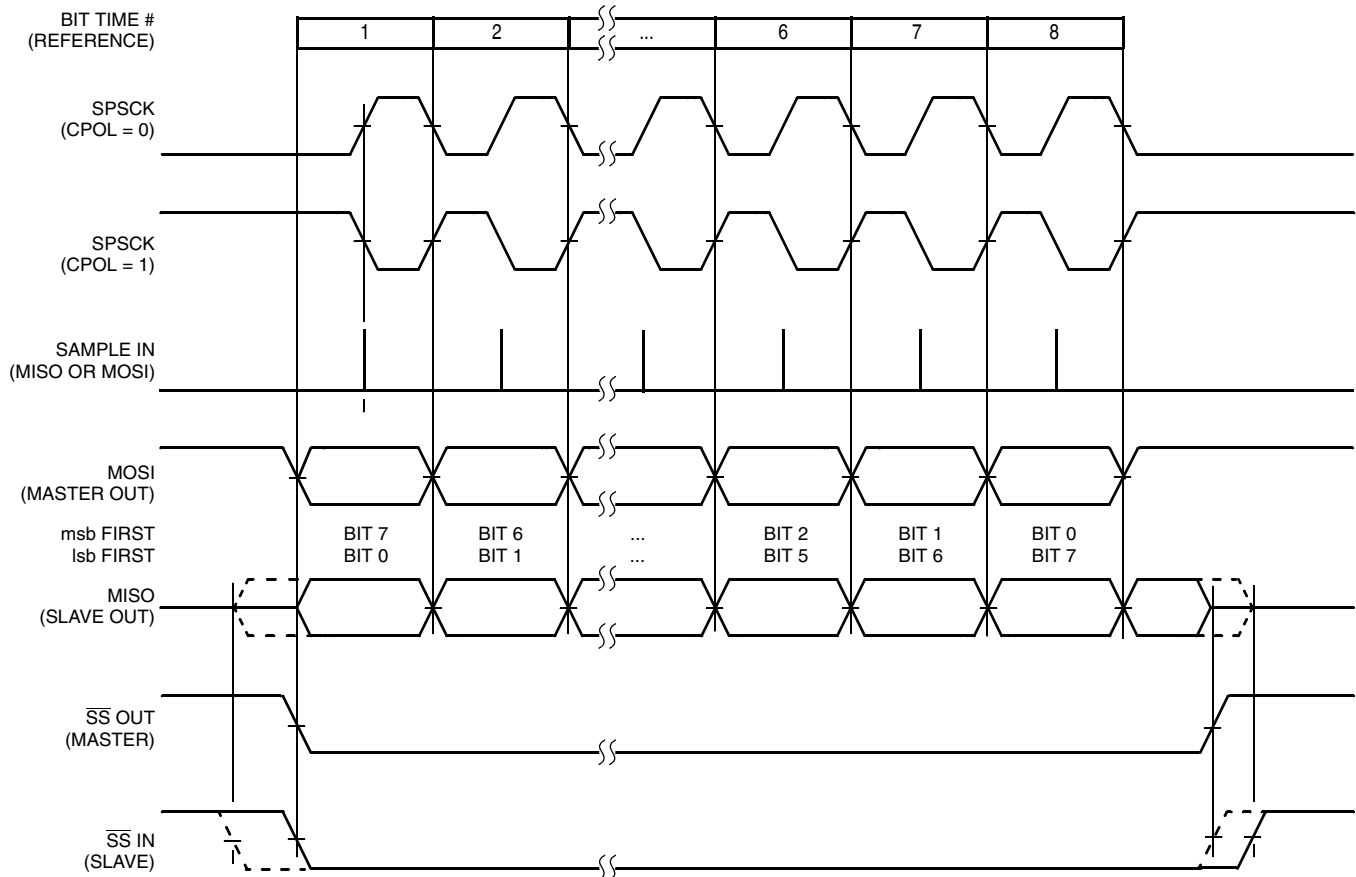**Figure 16-11. SPI Clock Formats (CPHA = 0)**

When CPHA is cleared, the slave begins to drive its MISO output with the first data bit value (msb or lsb depending on LSBFE) when $\overline{SS}$ asserts. The first SPSCK edge causes the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the second SPSCK edge, the SPI shifter shifts one bit position that shifts in the bit value that was sampled and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When CPHA is cleared, the slave's $\overline{SS}$ input must negate between transfers.

16.5.2   SPI Interrupts

There are three flag bits, two interrupt mask bits, and one interrupt vector associated with the SPI system. The SPIE bit enables interrupts from the SPI receiver full flag (SPRF) and mode fault flag (MODF). The SPTIE bit enables interrupts from the SPI transmit buffer empty flag (SPTEF). When one of the flag bits is set, and the associated interrupt mask bit is set, a hardware interrupt request is sent to the CPU. If the interrupt mask bits are cleared, software can poll the associated flag bits instead of using interrupts. The SPI interrupt service routine (ISR) should check the flag bits to determine what event caused the interrupt. The service routine should also clear the flag bit(s) before returning from the ISR (usually near the beginning of the ISR).

16.5.3   Mode Fault Detection

A mode fault occurs and the mode fault flag (MODF) sets when a master SPI device detects an error on the $\overline{SS}$ pin (provided the $\overline{SS}$ pin is configured as the mode fault input signal). The $\overline{SS}$ pin is configured as the mode fault input signal when MSTR and MODFEN is set, and SSOE is clear.

The mode fault detection feature is used in a system where more than one SPI device might become a master at the same time. The error is detected when a master's $\overline{SS}$ pin is low, indicating that some other SPI device is trying to address this master as if it were a slave. This could indicate a harmful output driver conflict, so the mode fault logic is designed to disable all SPI output drivers when such an error is detected.

When a mode fault is detected, MODF is set and MSTR is cleared to change the SPI configuration back to slave mode. The output drivers on the SPSCK, MOSI, and MISO (if not bidirectional mode) are disabled.

MODF is cleared by reading it while it is set, then writing to the SPIxC1 register. Software should verify the error condition has been corrected before changing the SPI back to master mode.

Chapter 17
Timer/Pulse-Width Modulator (S08TPMV3)

17.1 Introduction

Figure 17-1 shows the MCF51QE128 Series block diagram with the TPM highlighted.

17.1.1 ACMP/TPM Configuration Information

The ACMP modules can be configured to connect the output of the analog comparator to a TPM input capture channel 0 by setting the corresponding SOPT2[ACICx] bit. With ACICx set, the TPMxCH0 pin is not available externally regardless of the configuration of the TPMx module.

The ACMP1 output can be connected to TPM1CH0; The ACMP2 output can be connected to TPM2CH0.

17.1.2 TPM Clock Gating

The bus clock to TPM1, TPM2, and TPM3 can be gated on and off using the SCGC1[TPMx] bits. These bits are set after any reset, which enables the bus clock to this module. To conserve power, these bits can be cleared to disable the clock to any of these modules when not in use. See Section 5.6, "Peripheral Clock Gating," for details.

17.1.3 Interrupt Vector

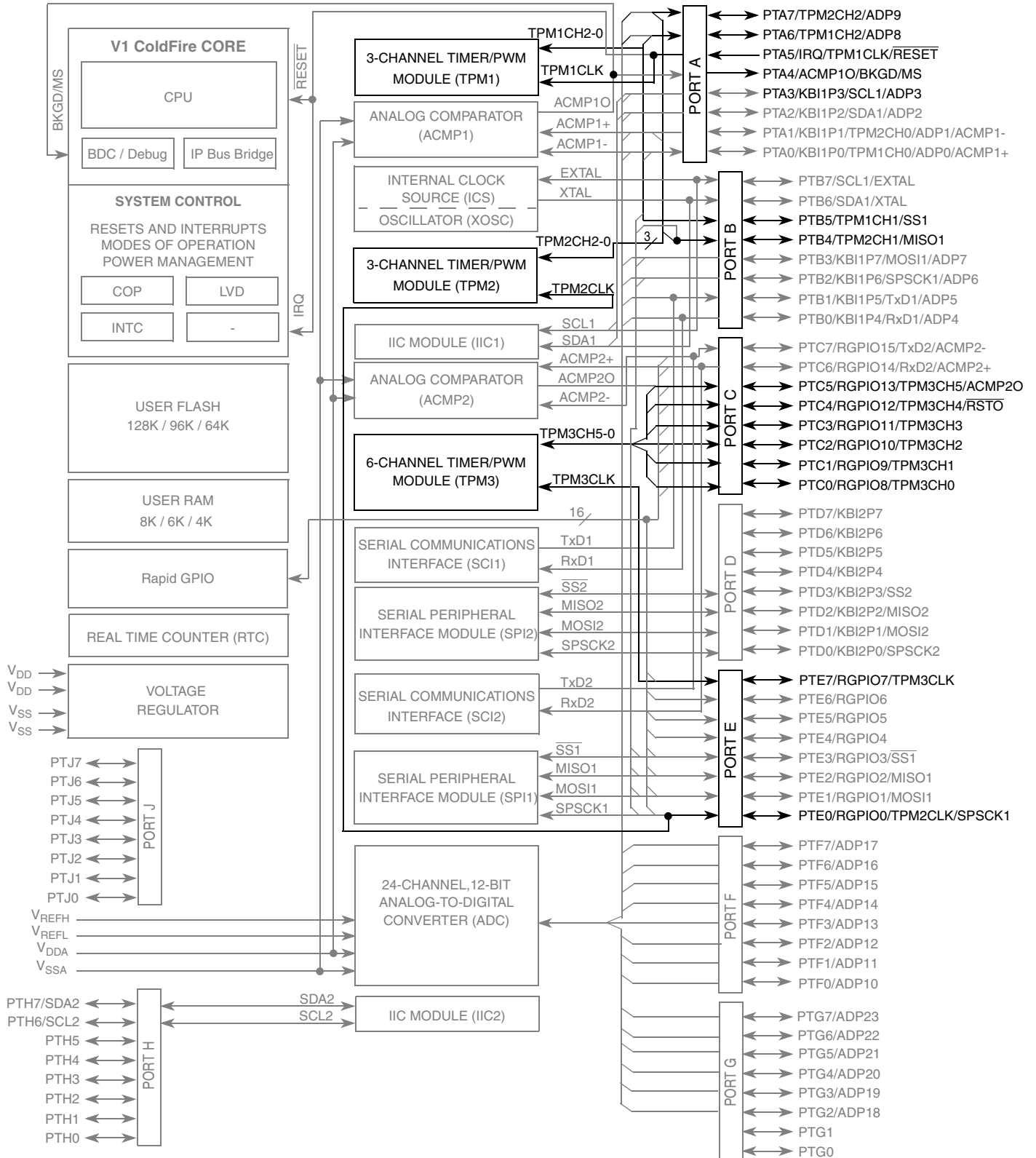See Chapter 8, "Interrupt Controller (CF1_INTC)," for the TPM interrupt vector assignments.

Figure 17-1. MCF51QE128 Series Block Diagram Highlighting TPM Block and Pins

Get the latest version from freescale.com

17.1.4   Features

The TPM includes these distinctive features:

- One to eight channels:
    - — Each channel may be input capture, output compare, or edge-aligned PWM
    - — Rising-Edge, falling-edge, or any-edge input capture trigger
    - — Set, clear, or toggle output compare action
    - — Selectable polarity on PWM outputs
- Module may be configured for buffered, center-aligned pulse-width-modulation (CPWM) on all channels
- Timer clock source selectable as prescaled bus clock, fixed system clock, or an external clock pin
    - — Prescale taps for divide-by 1, 2, 4, 8, 16, 32, 64, or 128
    - — Fixed system clock source are synchronized to the bus clock by an on-chip synchronization circuit
    - — External clock pin may be shared with any timer channel pin or a separated input pin
- 16-bit free-running or modulo up/down count operation
- Timer system enable
- One interrupt per channel plus terminal count interrupt

17.1.5   Modes of Operation

In general, TPM channels may be independently configured to operate in input capture, output compare, or edge-aligned PWM modes. A control bit allows the whole TPM (all channels) to switch to center-aligned PWM mode. When center-aligned PWM mode is selected, input capture, output compare, and edge-aligned PWM functions are not available on any channels of this TPM module.

When the microcontroller is in active BDM background or BDM foreground mode, the TPM temporarily suspends all counting until the microcontroller returns to normal user operating mode. During stop mode, all system clocks, including the main oscillator, are stopped. Therefore, the TPM is effectively disabled until clocks resume. During wait mode, the TPM continues to operate normally. Provided the TPM does not need to produce a real time reference or provide the interrupt source(s) needed to wake the MCU from wait mode, you can save power by disabling TPM functions before entering wait mode.

- Input capture mode

    When a selected edge event occurs on the associated MCU pin, the current value of the 16-bit timer counter is captured into the channel value register and an interrupt flag bit is set. Rising edges, falling edges, any edge, or no edge (disable channel) may be selected as the active edge that triggers the input capture.

- Output compare mode

    When the value in the timer counter register matches the channel value register, an interrupt flag bit is set and a selected output action is forced on the associated MCU pin. The output compare action may be selected to force the pin to zero, force the pin to one, toggle the pin, or ignore the pin (used for software timing functions).

- Edge-aligned PWM mode

    The value of a 16-bit modulo register plus 1 sets the period of the PWM output signal. The channel value register sets the duty cycle of the PWM output signal. You may also choose the polarity of the PWM output signal. Interrupts are available at the end of the period and at the duty-cycle transition point. This type of PWM signal is called edge-aligned because the leading edges of all PWM signals are aligned with the beginning of the period, which is the same for all channels within a TPM.

- Center-aligned PWM mode

    Twice the value of a 16-bit modulo register sets the period of the PWM output, and the channel-value register sets the half-duty-cycle duration. The timer counter counts up until it reaches the modulo value and then counts down until it reaches zero. As the count matches the channel value register while counting down, the PWM output becomes active. When the count matches the channel value register while counting up, the PWM output becomes inactive. This type of PWM signal is called center-aligned because the centers of the active duty cycle periods for all channels are aligned with a count value of zero. This type of PWM is required for types of motors used in small appliances.

This is a high-level description only. Detailed descriptions of operating modes are in later sections.

17.1.6   Block Diagram

The TPM uses one input/output (I/O) pin per channel, TPMxCHn (timer channel n), where n is the channel number (1-8). The TPM shares its I/O pins with general purpose I/O port pins (refer to I/O pin descriptions in full-chip specification for the specific chip implementation).

Figure 17-2 shows the TPM structure. The central component of the TPM is the 16-bit counter that can operate as a free-running counter or a modulo up/down counter. The TPM counter (when operating in normal up-counting mode) provides the timing reference for the input capture, output compare, and edge-aligned PWM functions. The timer counter modulo registers, TPMxMODH:TPMxMODL, control the modulo value of the counter (the values 0x0000 or 0xFFFF effectively make the counter free running). Software can read the counter value at any time without affecting the counting sequence. Any write to either half of the TPMxCNT counter resets the counter, regardless of the data value written.
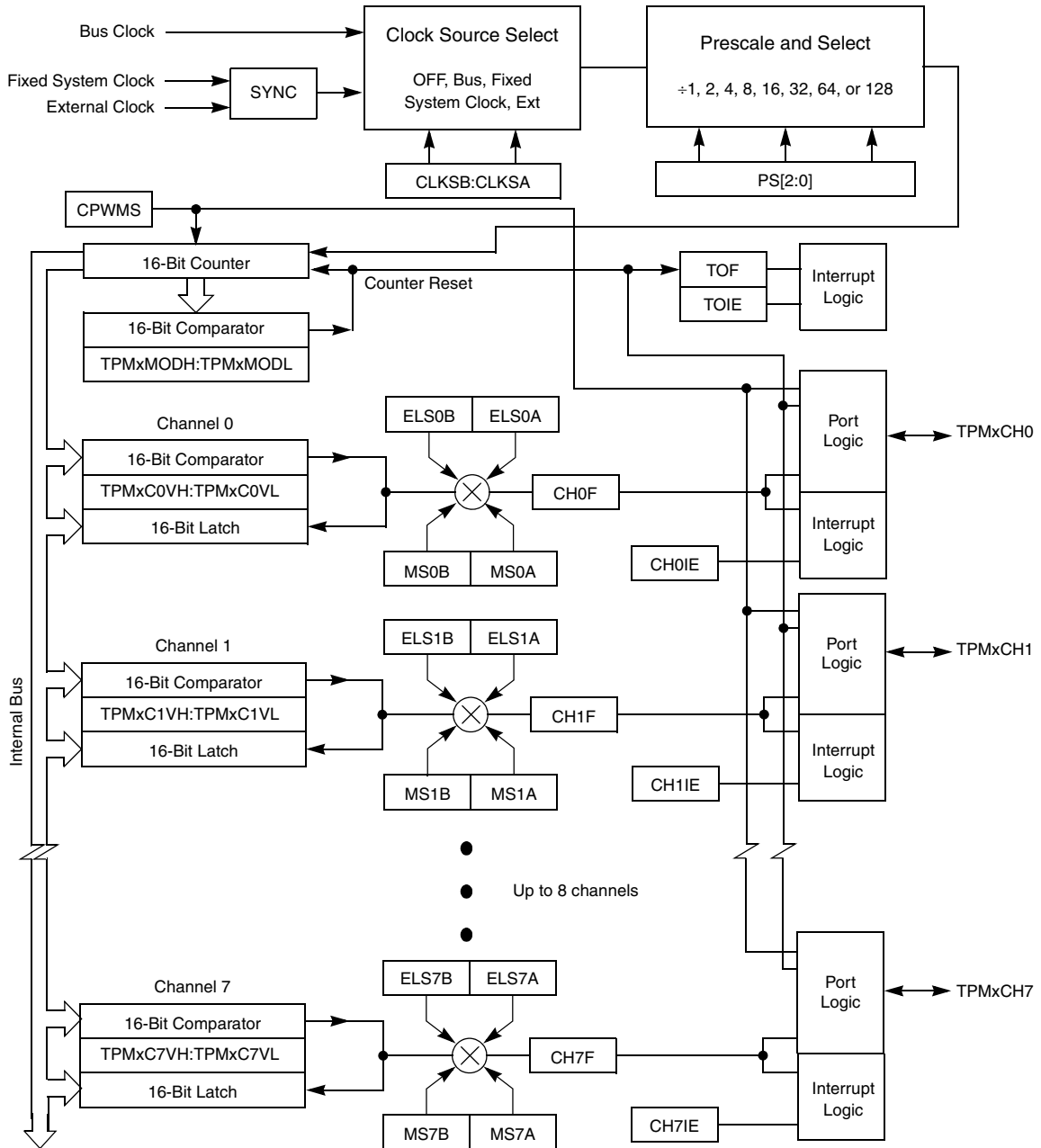
Figure 17-2. TPM Block Diagram

The TPM channels are programmable independently as input capture, output compare, or edge-aligned PWM channels. Alternately, the TPM can be configured to produce CPWM outputs on all channels. When the TPM is configured for CPWMs, the counter operates as an up/down counter; input capture, output compare, and EPWM functions are not practical.

If a channel is configured as input capture, an internal pullup device may be enabled for that channel. The details of how a module interacts with pin controls depends upon the chip implementation because the I/O pins and associated general purpose I/O controls are not part of the module. Refer to the discussion of the I/O port logic in a full-chip specification.

Because center-aligned PWMs are usually used to drive 3-phase AC-induction motors and brushless DC motors, they are typically used in sets of three or six channels.

17.2   Signal Description

Table 17-1 shows the user-accessible signals for the TPM. The number of channels may be varied from one to eight. When an external clock is included, it can be shared with the same pin as any TPM channel; however, it could be connected to a separate input pin. Refer to the I/O pin descriptions in full-chip specification for the specific chip implementation.

Table 17-1. Signal Properties

Name	Function
EXTCLK[1]	External clock source which may be selected to drive the TPM counter.
TPMxCHn[2]	I/O pin associated with TPM channel n

[1]   When preset, this signal can share any channel pin. However, depending upon full-chip implementation, this signal could be connected to a separate external pin.

[2]   n = channel number (1–8)

Refer to documentation for the full-chip for details about reset states, port connections, and whether there is any pullup device on these pins.

TPM channel pins can be associated with general purpose I/O pins and have passive pullup devices that can be enabled with a control bit when the TPM or general purpose I/O controls have configured the associated pin as an input. When no TPM function is enabled to use a corresponding pin, the pin reverts to being controlled by general purpose I/O controls, including the port-data and data-direction registers. Immediately after reset, no TPM functions are enabled, so all associated pins revert to general purpose I/O control.

17.2.1   Detailed Signal Descriptions

This section describes each user-accessible pin signal in detail. Although Table 17-1 grouped all channel pins together, any TPM pin can be shared with the external clock source signal. Because I/O pin logic is not part of the TPM, refer to full-chip documentation for a specific derivative for more details about the interaction of TPM pin functions and general purpose I/O controls including port data, data direction, and pullup controls.

17.2.1.1 EXTCLK — External Clock Source

Control bits in the timer status and control register allow you to select nothing (timer disable), the bus-rate clock (the normal default source), a crystal-related clock, or an external clock as the clock that drives the TPM prescaler and subsequently the 16-bit TPM counter. The external clock source is synchronized in the TPM. The bus clock clocks the synchronizer. The frequency of the external source must be no more than one-fourth the frequency of the bus-rate clock to meet Nyquist criteria and allowing for jitter.

The external clock signal shares the same pin as a channel I/O pin, so the channel pin is not usable for channel I/O function when selected as the external clock source. It is the your responsibility to avoid such settings. If this pin is used as an external clock source (CLKSB:CLKSA = 1:1), the channel can remain in output compare mode as a software timer (ELSnB:ELSnA = 0:0).

17.2.1.2 TPMxCHn — TPM Channel n I/O Pin(s)

Each TPM channel is associated with an I/O pin on the MCU. The function of this pin depends on the channel configuration. The TPM pins share with general purpose I/O pins, where each pin has a port data register bit, and a data direction control bit, and the port has optional passive pullups which may be enabled when a port pin is acting as an input.

The TPM channel does not control the I/O pin when ELSnB:ELSnA or CLKSB:CLKSA are cleared so it normally reverts to general purpose I/O control. When CPWMS is set and ELSnB:ELSnA are not cleared, all channels within the TPM are configured for center-aligned PWM and the TPMxCHn pins are all controlled by the TPM system. When CPWMS is cleared, the MSnB:MSnA control bits determine whether the channel is configured for input capture, output compare, or edge-aligned PWM.

When a channel is configured for input capture (CPWMS=0, MSnB:MSnA = 0:0 and ELSnB:ELSnA not = 0:0), the TPMxCHn pin is forced to act as an edge-sensitive input to the TPM. ELSnB:ELSnA control bits determine what polarity edge or edges trigger input-capture events. A synchronizer based on the bus clock synchronizes input edges to the bus clock. This implies the minimum pulse width—that can be reliably detected—on an input capture pin is four bus clock periods (with ideal clock pulses as near as two bus clocks can be detected). TPM uses this pin as an input capture input to override the port data and data direction controls for the same pin.

When a channel is configured for output compare (CPWMS = 0, MSnB:MSnA = 01 and ELSnB:ELSnA ≠ 00), the associated data direction control is overridden, the TPMxCHn pin is considered an output controlled by the TPM, and the ELSnB:ELSnA control bits determine how the pin is controlled. The remaining three combinations of ELSnB:ELSnA determine whether the TPMxCHn pin is toggled, cleared, or set each time the 16-bit channel value register matches the timer counter.

When the output compare toggle mode is initially selected, the previous value on the pin is driven out until the next output compare event, then the pin is toggled.

When a channel is configured for edge-aligned PWM (CPWMS = 0, MSnB = 1, ELSnB:ELSnA ≠ 00), the data direction is overridden, the TPMxCHn pin is forced to be an output controlled by the TPM, and ELSnA controls the polarity of the PWM output signal on the pin. When ELSnB is set and ELSnA is cleared, the TPMxCHn pin is forced high at the start of each new period (TPMxCNT=0x0000). The pin is forced low when the channel value register matches the timer counter. When ELSnA is set, the TPMxCHn

pin is forced low at the start of each new period (TPMxCNT=0x0000), and the pin is forced high when the channel value register matches the timer counter.

TPMxMODH:TPMxMODL = 0x0008
TPMxMODH:TPMxMODL = 0x0005
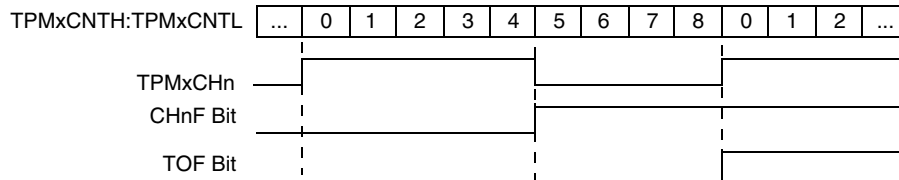
Figure 17-3. High-True Pulse of an Edge-Aligned PWM

TPMxMODH:TPMxMODL = 0x0008
TPMxMODH:TPMxMODL = 0x0005
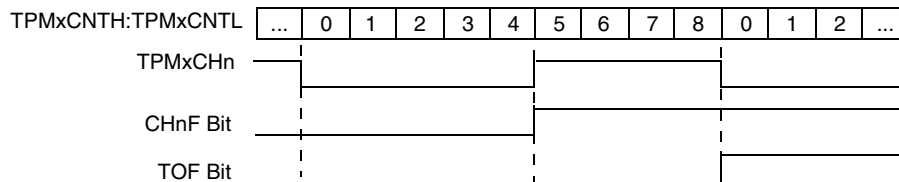
Figure 17-4. Low-True Pulse of an Edge-Aligned PWM

When the TPM is configured for center-aligned PWM and ELSnB:ELSnA are not cleared, the data direction for all channels in this TPM are overridden, the TPMxCHn pins are forced to be outputs controlled by the TPM, and the ELSnA bits control the polarity of each TPMxCHn output. If ELSnB is set and ELSnA is cleared, the corresponding TPMxCHn pin is negated when the timer counter is counting up, and the channel value register matches the timer counter. The TPMxCHn pin is asserted when the timer counter is counting down, and the channel value register matches the timer counter. If ELSnA is set, the corresponding TPMxCHn pin is asserted when the timer counter is counting up and the channel value register matches the timer counter; the TPMxCHn pin is negated when the timer counter is counting down and the channel value register matches the timer counter.

TPMxMODH:TPMxMODL = 0x0008
TPMxMODH:TPMxMODL = 0x0005
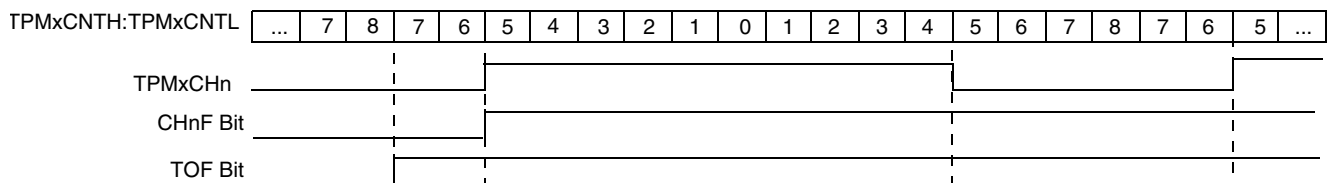
Figure 17-5. High-True Pulse of a Center-Aligned PWM

TPMxMODH:TPMxMODL = 0x0008
TPMxMODH:TPMxMODL = 0x0005
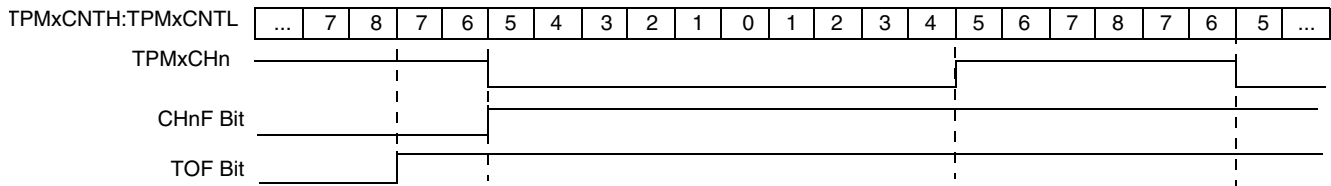
Figure 17-6. Low-True Pulse of a Center-Aligned PWM

17.3   Register Definition

This section consists of register descriptions in address order.

17.3.1   TPM Status and Control Register (TPMxSC)

TPMxSC contains the overflow status flag and control bits used to configure the interrupt enable, TPM configuration, clock source, and prescale factor. These controls relate to all channels within this timer module.

Figure 17-7. TPM Status and Control Register (TPMxSC)

Table 17-2. TPMxSC Field Descriptions

Field	Description
7 TOF	Timer overflow flag. This read/write flag is set when the TPM counter resets to 0x0000 after reaching the modulo value programmed in the TPM counter modulo registers. Clear TOF by reading the TPM status and control register when TOF is set and then writing a logic 0 to TOF. If another TPM overflow occurs before the clearing sequence is complete, the sequence is reset so TOF would remain set after the clear sequence was completed for the earlier TOF. This is done so a TOF interrupt request cannot be lost during the clearing sequence for a previous TOF. Reset clears TOF. Writing a logic 1 to TOF has no effect. 0 TPM counter has not reached modulo value or overflow 1 TPM counter has overflowed
6 TOIE	Timer overflow interrupt enable. This read/write bit enables TPM overflow interrupts. If TOIE is set, an interrupt is generated when TOF equals one. Reset clears TOIE. 0 TOF interrupts inhibited (use for software polling) 1 TOF interrupts enabled

Table 17-2. TPMxSC Field Descriptions (continued)

Field	Description
5 CPWMS	Center-aligned PWM select. When present, this read/write bit selects CPWM operating mode. By default, the TPM operates in up-counting mode for input capture, output compare, and edge-aligned PWM functions. Setting CPWMS reconfigures the TPM to operate in up/down counting mode for CPWM functions. Reset clears CPWMS. 0 All channels operate as input capture, output compare, or edge-aligned PWM mode as selected by the MSnB:MSnA control bits in each channel's status and control register. 1 All channels operate in center-aligned PWM mode.
4–3 CLKS[B:A]	Clock source selects. As shown in Table 17-3, this 2-bit field disables the TPM system or selects one of three clock sources to drive the counter prescaler. The fixed system clock source is only meaningful in systems with a PLL-based system clock. When there is no PLL, the fixed-system clock source is the same as the bus rate clock. The external source is synchronized to the bus clock by TPM module, and the fixed system clock source (when a PLL is present) is synchronized to the bus clock by an on-chip synchronization circuit. When a PLL is present but not enabled, the fixed-system clock source is the same as the bus-rate clock.
2–0 PS[2:0]	Prescale factor select. This 3-bit field selects one of eight division factors for the TPM clock input as shown in Table 17-4. This prescaler is located after any clock source synchronization or clock source selection so it affects the clock source selected to drive the TPM system. The new prescale factor affects the clock source on the next system clock cycle after the new value is updated into the register bits.

Table 17-3. TPM-Clock-Source Selection

CLKSB:CLKSA	TPM Clock Source to Prescaler Input
00	No clock selected (TPM counter disable)
01	Bus rate clock
10	Fixed system clock
11	External source

Table 17-4. Prescale Factor Selection

PS[2:0]	TPM Clock Source Divided-by
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

17.3.2 TPM Counter Registers (TPMxCNTH:TPMxCNTL)

The two read-only TPM counter registers contain the high and low bytes of the value in the TPM counter. Reading either byte (TPMxCNTH or TPMxCNTL) latches the contents of both bytes into a buffer where they remain latched until the other half is read. This allows coherent 16-bit reads in big-endian or

---

MCF51QE128 MCU Series Reference Manual, Rev. 3

little-endian order that makes this more friendly to various compiler implementations. The coherency mechanism is automatically restarted by an MCU reset or any write to the timer status/control register (TPMxSC).

Reset clears the TPM counter registers. Writing any value to TPMxCNTH or TPMxCNTL also clears the TPM counter (TPMxCNTH:TPMxCNTL) and resets the coherency mechanism, regardless of the data involved in the write.

	7	6	5	4	3	2	1	0
R				TPMxCNT[15:8]				
W				Any write to TPMxCNTH clears the 16-bit counter				
Reset	0	0	0	0	0	0	0	0
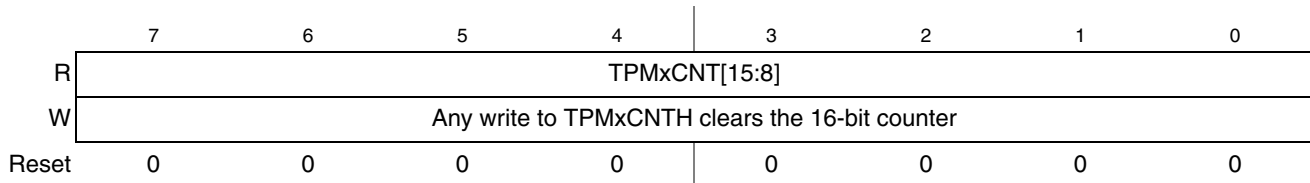
Figure 17-8. TPM Counter Register High (TPMxCNTH)

	7	6	5	4	3	2	1	0
R				TPMxCNT[7:0]				
W				Any write to TPMxCNTL clears the 16-bit counter				
Reset	0	0	0	0	0	0	0	0
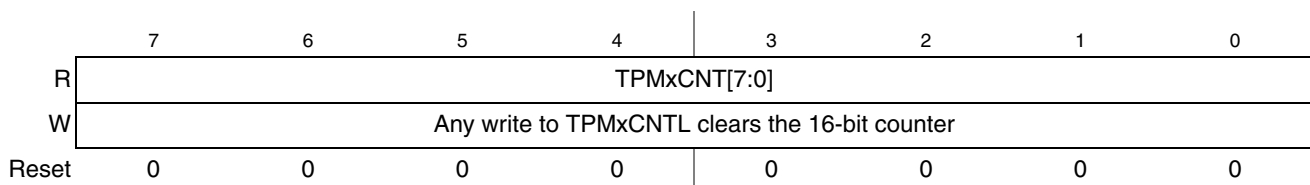
Figure 17-9. TPM Counter Register Low (TPMxCNTL)

When BDM is active, the timer counter is frozen (this is the value read by user). The coherency mechanism is frozen so the buffer latches remain in the state they were in when the BDM became active, even if one or both counter halves are read while BDM is active. This assures that if the user was in the middle of reading a 16-bit register when BDM became active, it reads the appropriate value from the other half of the 16-bit value after returning to normal execution.

In BDM mode, writing any value to TPMxSC, TPMxCNTH, or TPMxCNTL registers resets the read coherency mechanism of the TPMxCNTH:L registers, regardless of the data involved in the write.

17.3.3 TPM Counter Modulo Registers (TPMxMODH:TPMxMODL)

The read/write TPM modulo registers contain the modulo value for the TPM counter. After the TPM counter reaches the modulo value, the TPM counter resumes counting from 0x0000 at the next clock, and the overflow flag (TOF) becomes set. Writing to TPMxMODH or TPMxMODL inhibits the TOF bit and overflow interrupts until the other byte is written. Reset sets the TPM counter modulo registers to 0x0000 which results in a free running timer counter (modulo disabled).

Writing to either byte (TPMxMODH or TPMxMODL) latches the value into a buffer and the registers are updated with the value of their write buffer according to the value of CLKSB:CLKSA bits, so:

- If CLKSB and CLKSA are cleared, the registers are updated when the second byte is written
- If CLKSB and CLKSA are not cleared, the registers are updated after both bytes were written, and the TPM counter changes from (TPMxMODH:TPMxMODL - 1) to (TPMxMODH:TPMxMODL). If the TPM counter is a free-running counter, the update is made when the TPM counter changes from 0xFFFE to 0xFFFF

The latching mechanism may be manually reset by writing to the TPMxSC address (whether BDM is active or not).

When BDM is active, the coherency mechanism is frozen (unless reset by writing to TPMxSC) so the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the modulo register are written while BDM is active. Any write to the modulo registers bypasses the buffer latches and directly writes to the modulo register while BDM is active.

	7	6	5	4	3	2	1	0
R								
W				TPMxMOD[15:8]				
Reset	0	0	0	0	0	0	0	0
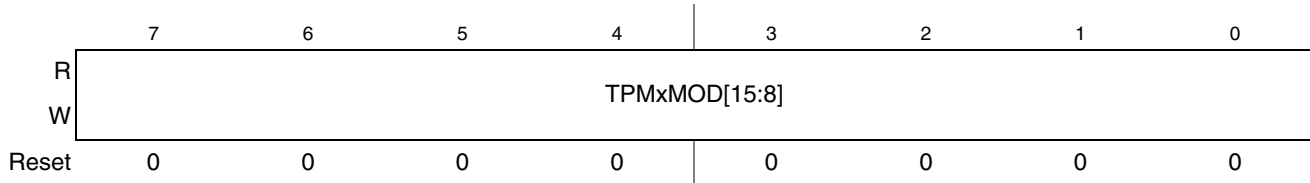
Figure 17-10. TPM Counter Modulo Register High (TPMxMODH)

	7	6	5	4	3	2	1	0
R								
W				TPMxMOD[7:0]				
Reset	0	0	0	0	0	0	0	0
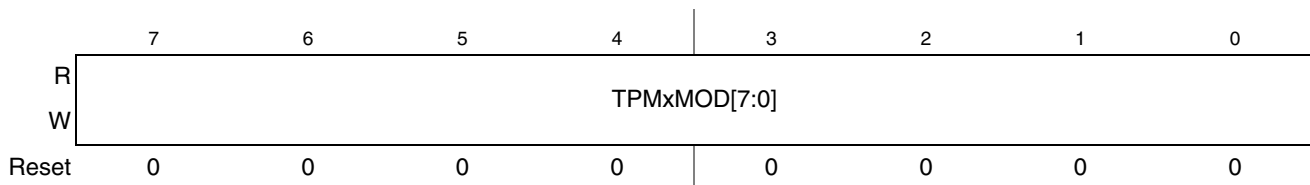
Figure 17-11. TPM Counter Modulo Register Low (TPMxMODL)

Reset the TPM counter before writing to the TPM modulo registers to avoid confusion about when the first counter overflow occurs.

17.3.4   TPM Channel n Status and Control Register (TPMxCnSC)

TPMxCnSC contains the channel-interrupt-status flag and control bits that configure the interrupt enable, channel configuration, and pin function.

	7	6	5	4	3	2	1	0
R	CHnF	CHnIE	MSnB	MSnA	ELSnB	ELSnA	0	0
W	0							
Reset	0	0	0	0	0	0	0	0
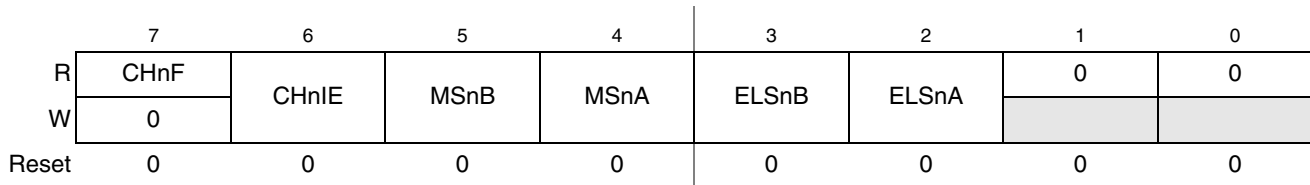
Figure 17-12. TPM Channel n Status and Control Register (TPMxCnSC)

Table 17-5. TPMxCnSC Field Descriptions

Field	Description
7 CHnF	Channel n flag. When channel n is an input-capture channel, this read/write bit is set when an active edge occurs on the channel n pin. When channel n is an output compare or edge-aligned/center-aligned PWM channel, CHnF is set when the value in the TPM counter registers matches the value in the TPM channel n value registers. When channel n is an edge-aligned/center-aligned PWM channel and the duty cycle is set to 0% or 100%, CHnF is not set even when the value in the TPM counter registers matches the value in the TPM channel n value registers. A corresponding interrupt is requested when this bit is set and interrupts are enabled (CHnIE = 1). Clear CHnF by reading TPMxCnSC while this bit is set and then writing a logic 0 to it. If another interrupt request occurs before the clearing sequence is complete, CHnF remains set. This is done so a CHnF interrupt request is not lost due to clearing a previous CHnF. Reset clears this bit. Writing a logic 1 to CHnF has no effect. 0　No input capture or output compare event occurred on channel n 1　Input capture or output compare event on channel n
6 CHnIE	Channel n interrupt enable. This read/write bit enables interrupts from channel n. Reset clears this bit. 0　Channel n interrupt requests disabled (use for software polling) 1　Channel n interrupt requests enabled
5 MSnB	Mode select B for TPM channel n. When CPWMS is cleared, setting this bit configures TPM channel n for edge-aligned PWM mode. Refer to the summary of channel mode and setup controls in Table 17-6.
4 MSnA	Mode select A for TPM channel n. When CPWMS and MSnB are cleared, MSnA configures TPM channel n for input-capture mode or output compare mode. Refer to Table 17-6 for a summary of channel mode and setup controls. **Note:** If the associated port pin is not stable for at least two bus clock cycles before changing to input capture mode, it is possible to get an unexpected indication of an edge trigger.
3–2 ELSnB ELSnA	Edge/level select bits. Depending upon the operating mode for the timer channel as set by CPWMS:MSnB:MSnA and shown in Table 17-6, these bits select the polarity of the input edge that triggers an input capture event, select the level that is driven in response to an output compare match or select the polarity of the PWM output. Clearing these bits configures the related timer pin as a general purpose I/O pin. This function is typically used to temporarily disable an input capture channel or to make the timer pin available as a general purpose I/O pin when the associated timer channel is set up as a software timer that does not require the use of a pin.

Table 17-6. Mode, Edge, and Level Selection

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	XX	00	Pin not used for TPM. Revert to GPIO or other peripheral control	
0	00	01	Input capture	Capture on rising edge only
		10		Capture on falling edge only
		11		Capture on rising or falling edge
	01	01	Output compare	Toggle output on compare
		10		Clear output on compare
		11		Set output on compare
	1X	10	Edge-aligned PWM	High-true pulses (clear output on compare)
		X1		Low-true pulses (set output on compare)

Table 17-6. Mode, Edge, and Level Selection (continued)

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
1	XX	10	Center-aligned PWM	High-true pulses (clear output on compare-up)
		X1		Low-true pulses (set output on compare-up)

17.3.5   TPM Channel Value Registers (TPMxCnVH:TPMxCnVL)

These read/write registers contain the captured TPM counter value of the input capture function or the output compare value for the output compare or PWM functions. The channel registers are cleared by reset.

	7	6	5	4	3	2	1	0
R								
W				TPMxCnV[15:8]				
Reset	0	0	0	0	0	0	0	0
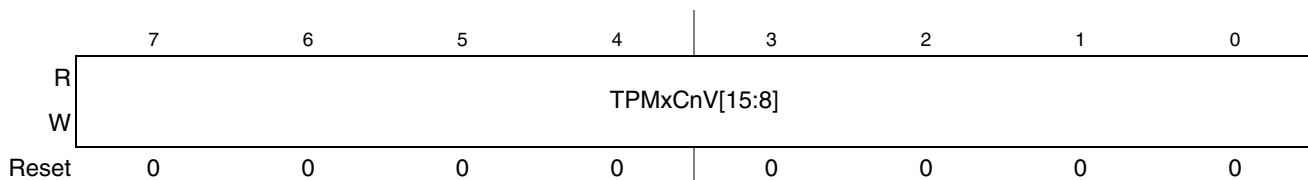
Figure 17-13. TPM Channel Value Register High (TPMxCnVH)

	7	6	5	4	3	2	1	0
R								
W				TPMxCnV[7:0]				
Reset	0	0	0	0	0	0	0	0
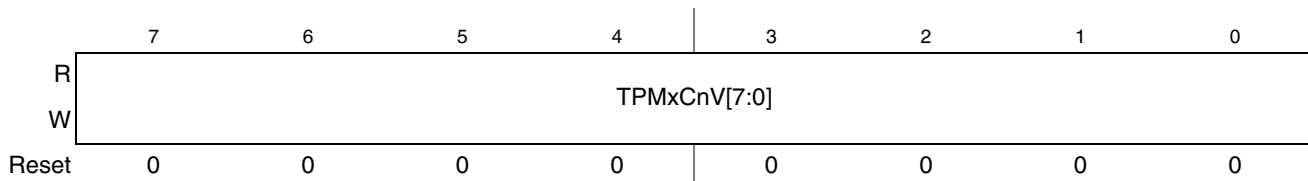
Figure 17-14. TPM Channel Value Register Low (TPMxCnVL)

In input capture mode, reading either byte (TPMxCnVH or TPMxCnVL) latches the contents of both bytes into a buffer where they remain latched until the other half is read. This latching mechanism also resets (becomes unlatched) when the TPMxCnSC register is written (whether BDM mode is active or not). Any write to the channel registers is ignored during the input capture mode.

When BDM is active, the coherency mechanism is frozen (unless reset by writing to TPMxCnSC register) so the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the channel register are read while BDM is active. This assures that if the user was in the middle of reading a 16-bit register when BDM became active, it reads the appropriate value from the other half of the 16-bit value after returning to normal execution. The value read from the TPMxCnVH and TPMxCnVL registers in BDM mode is the value of these registers and not the value of their read buffer.

In output compare or PWM modes, writing to either byte (TPMxCnVH or TPMxCnVL) latches the value into a buffer. After both bytes are written, they are transferred as a coherent 16-bit value into the timer-channel registers according to the value of CLKSB:CLKSA bits and the selected mode, so:

- If CLKSB and CLKSA are cleared, the registers are updated when the second byte is written.
- If CLKSB and CLKSA are not cleared and in output compare mode, the registers are updated after the second byte is written and on the next change of the TPM counter (end of the prescaler counting).

MCF51QE128 MCU Series Reference Manual, Rev. 3

- If CLKSB and CLKSA are not cleared and in EPWM or CPWM modes, the registers are updated after the both bytes were written, and the TPM counter changes from TPMxMODH:TPMxMODL − 1 to TPMxMODH:TPMxMODL. If the TPM counter is a free-running counter, the update is made when the TPM counter changes from 0xFFFE to 0xFFFF.

The latching mechanism may be manually reset by writing to the TPMxCnSC register (whether BDM mode is active or not). This latching mechanism allows coherent 16-bit writes in big-endian or little-endian order that is friendly to various compiler implementations.

When BDM is active, the coherency mechanism is frozen so the buffer latches remain in the state they were in when the BDM became active even if one or both halves of the channel register are written while BDM is active. Any write to the channel registers bypasses the buffer latches and directly write to the channel register while BDM is active. The values written to the channel register while BDM is active are used for PWM and output compare operation after normal execution resumes. Writes to the channel registers while BDM is active do not interfere with partial completion of a coherency sequence. After the coherency mechanism has been fully exercised, the channel registers are updated using the buffered values written (while BDM was not active) by the user.

17.4   Functional Description

All TPM functions are associated with a central 16-bit counter that allows flexible selection of the clock source and prescale factor. There is also a 16-bit modulo register associated with the main counter.

The CPWMS control bit chooses between center-aligned PWM operation for all channels in the TPM (CPWMS=1) or general purpose timing functions (CPWMS=0) where each channel can independently be configured to operate in input capture, output compare, or edge-aligned PWM mode. The CPWMS control bit is located in the main TPM status and control register because it affects all channels within the TPM and influences the way the main counter operates. (In CPWM mode, the counter changes to an up/down mode rather than the up-counting mode used for general purpose timer functions.)

The following sections describe the main counter and each of the timer operating modes (input capture, output compare, edge-aligned PWM, and center-aligned PWM). Because details of pin operation and interrupt activity depend upon the operating mode, these topics are covered in the associated mode explanation sections.

17.4.1   Counter

All timer functions are based on the main 16-bit counter (TPMxCNTH:TPMxCNTL). This section discusses selection of the clock source, end-of-count overflow, up-counting vs. up/down counting, and manual counter reset.

17.4.1.1   Counter Clock Source

The 2-bit field, CLKSB:CLKSA, in the timer status and control register (TPMxSC) selects one of three possible clock sources or OFF (which effectively disables the TPM). See Table 17-3. After any MCU reset, CLKSB and CLKSA are cleared so no clock source is selected, and the TPM is in a very low power state. These control bits may be read or written at any time and disabling the timer (writing 00 to the CLKSB:CLKSA field) does not affect the values in the counter or other timer registers.

Table 17-7. TPM Clock Source Selection

CLKSB:CLKSA	TPM Clock Source to Prescaler Input
00	No clock selected (TPM counter disabled)
01	Bus rate clock
10	Fixed system clock
11	External source

The bus rate clock is the main system bus clock for the MCU. This clock source requires no synchronization because it is the clock used for all internal MCU activities including operation of the CPU and buses.

In MCUs that have no PLL or the PLL is not engaged, the fixed system clock source is the same as the bus-rate-clock source, and it does not go through a synchronizer. When a PLL is present and engaged, a synchronizer is required between the crystal divided-by-two clock source and the timer counter. This means counter transitions are properly aligned to bus-clock transitions. A synchronizer is used at chip level to synchronize the crystal-related source clock to the bus clock.

The external clock source may be connected to any TPM channel pin. This clock source always has to pass through a synchronizer to assure that counter transitions are properly aligned to bus clock transitions. The bus-rate clock drives the synchronizer; therefore, to meet Nyquist criteria even with jitter, the frequency of the external clock source must not be faster than the bus rate divided-by four. With ideal clocks, the external clock can be as fast as bus clock divided by four.

When the external clock source shares the TPM channel pin, this pin should not be used for other channel timing functions. For example, it would be ambiguous to configure channel 0 for input capture when the TPM channel 0 pin was also being used as the timer external clock source. It is your responsibility to avoid such settings. The TPM channel could be used in output compare mode for software timing functions (pin controls set not to affect the TPM channel pin).

17.4.1.2   Counter Overflow and Modulo Reset

An interrupt flag and enable are associated with the 16-bit main counter. The flag (TOF) is a software-accessible indication that the timer counter has overflowed. The enable signal selects between software polling (TOIE = 0) where no hardware interrupt is generated or interrupt-driven operation (TOIE = 1) where a static hardware interrupt is generated whenever TOF is set.

The conditions causing TOF to become set depend on whether the TPM is configured for center-aligned PWM (CPWMS = 1). In the simplest mode, there is no modulus limit and the TPM is not in center-aligned PWM mode. In this case, the 16-bit timer counter counts from 0x0000 through 0xFFFF and overflows to 0x0000 on the next counting clock. TOF is set at the transition from 0xFFFF to 0x0000. When a modulus limit is set, TOF is set at the transition from the value set in the modulus register to 0x0000. When the TPM is in center-aligned PWM mode (CPWMS = 1), the TOF flag is set as the counter changes direction at the end of the count value set in the modulus register (at the transition from the value set in the modulus register to the next lower count value). This corresponds to the end of a PWM period (the 0x0000 count value corresponds to the center of a period).

17.4.1.3 Counting Modes

The main timer counter has two counting modes. When center-aligned PWM is selected (CPWMS = 1), the counter operates in up/down counting mode. Otherwise, the counter operates as a simple up counter. As an up counter, the timer counter counts from 0x0000 through its terminal count and continues with 0x0000. The terminal count is 0xFFFF or a modulus value in TPMxMODH:TPMxMODL.

When center-aligned PWM operation is specified, the counter counts up from 0x0000 through its terminal count and then down to 0x0000 where it changes back to up counting. The terminal count value and 0x0000 are normal length counts (one timer clock period long). In this mode, the timer overflow flag (TOF) becomes set at the end of the terminal-count period as the count changes to the next lower count value.

17.4.1.4 Manual Counter Reset

The main timer counter can be manually reset at any time by writing any value to either TPMxCNTH or TPMxCNTL. Resetting the counter in this manner also resets the coherency mechanism in case only half of the counter was read before resetting the count.

17.4.2 Channel Mode Selection

Provided CPWMS is cleared, TPMxCnSC[MSnB,MSnA] determine the basic mode of operation for the corresponding channel. Choices include input capture, output compare, and edge-aligned PWM.

17.4.2.1 Input Capture Mode

With the input-capture function, the TPM can capture the time an external event occurs. When an active edge occurs on the pin of an input-capture channel, the TPM latches the contents of the TPM counter into the channel-value registers (TPMxCnVH:TPMxCnVL). Rising edges, falling edges, or any edge may be chosen as the active edge that triggers an input capture.

In input capture mode, TPMxCnVH and TPMxCnVL are read-only.

When either half of the 16-bit capture register is read, the other half is latched into a buffer to support coherent 16-bit accesses in big-endian or little-endian order. The coherency sequence can be manually reset by writing to TPMxCnSC.

An input capture event sets a flag bit (CHnF) that may optionally generate a CPU interrupt request.

While in BDM, the input capture function works as configured. When an external event occurs, the TPM latches the contents of the TPM counter (frozen because of the BDM mode) into the channel value registers and sets the flag bit.

17.4.2.2 Output Compare Mode

With the output-compare function, the TPM can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter reaches the value in the channel-value registers of an output-compare channel, the TPM can set, clear, or toggle the channel pin.

In output compare mode, values are transferred to the corresponding timer channel registers only after both 8-bit halves of a 16-bit register have been written and according to the value of CLKSB:CLKSA bits, so:

- If CLKSB:CLKSA are cleared, the registers are updated when the second byte is written
- If CLKSB:CLKSA are not cleared, the registers are updated at the next change of the TPM counter (end of the prescaler counting) after the second byte is written.

The coherency sequence can be manually reset by writing to the channel status/control register (TPMxCnSC).

An output compare event sets a flag bit (CHnF) which may optionally generate a CPU-interrupt request.

17.4.2.3   Edge-Aligned PWM Mode

This type of PWM output uses the normal up-counting mode of the timer counter (CPWMS=0) and can be used when other channels in the same TPM are configured for input capture or output compare functions. The period of this PWM signal is determined by the value of the modulus register (TPMxMODH:TPMxMODL) plus 1. The duty cycle is determined by the setting in the timer channel register (TPMxCnVH:TPMxCnVL). The polarity of this PWM signal is determined by the setting in the ELSnA control bit. 0% and 100% duty cycle cases are possible.

The output compare value in the TPM channel registers determines the pulse width (duty cycle) of the PWM signal (Figure 17-15). The time between the modulus overflow and the output compare is the pulse width. If ELSnA is cleared, the counter overflow forces the PWM signal high, and the output compare forces the PWM signal low. If ELSnA is set, the counter overflow forces the PWM signal low, and the output compare forces the PWM signal high.
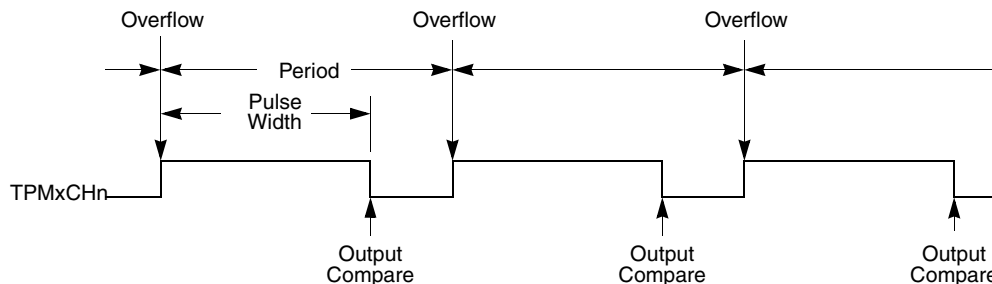
Figure 17-15. PWM Period and Pulse Width (ELSnA=0)

When the channel value register is set to 0x0000, the duty cycle is 0%. A 100% duty cycle is achieved by setting the timer-channel register (TPMxCnVH:TPMxCnVL) to a value greater than the modulus setting. This implies that the modulus setting must be less than 0xFFFF to get 100% duty cycle.

Since the TPM is connected to an 8-bit peripheral bus, the settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to any of the registers, TPMxCnVH and TPMxCnVL, actually writes to buffer registers. In edge-aligned PWM mode, values are transferred to the corresponding timer-channel registers according to the value of CLKSB:CLKSA bits:

- If CLKSB and CLKSA are cleared, the registers are updated when the second byte is written

- If CLKSB and CLKSA are not cleared, the registers are updated after both bytes are written, and the TPM counter changes from TPMxMODH:TPMxMODL − 1 to TPMxMODH:TPMxMODL. If the TPM counter is a free-running counter, the update is made when the TPM counter changes from 0xFFFE to 0xFFFF.

17.4.2.4   Center-Aligned PWM Mode

This type of PWM output uses the up/down counting mode of the timer counter (CPWMS=1). The output compare value in TPMxCnVH:TPMxCnVL determines the pulse width (duty cycle) of the PWM signal while the period is determined by the value in TPMxMODH:TPMxMODL. TPMxMODH:TPMxMODL should be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results. ELSnA determines the polarity of the CPWM output.

Pulse width = 2 × (TPMxCnVH:TPMxCnVL)

Period = 2 × (TPMxMODH:TPMxMODL); TPMxMODH:TPMxMODL = 0x0001–0x7FFF

If the channel-value register, TPMxCnVH:TPMxCnVL, is zero or negative (bit 15 set), the duty cycle is 0%. If TPMxCnVH:TPMxCnVL is a positive value (bit 15 clear) and is greater than the non-zero modulus setting, the duty cycle is 100% because the duty cycle compare never occurs. This implies the usable range of periods set by the modulus register is 0x0001 through 0x7FFE (0x7FFF if you do not need to generate 100% duty cycle). This is not a significant limitation. The resulting period would be much longer than required for normal applications.

All zeros in TPMxMODH:TPMxMODL is a special case that should not be used with center-aligned PWM mode. When CPWMS is cleared, this case corresponds to the counter running free from 0x0000 through 0xFFFF. When CPWMS is set, the counter needs a valid match to the modulus register somewhere other than at 0x0000 in order to change directions from up-counting to down-counting.

The output compare value in the TPM channel registers (times two) determines the pulse width (duty cycle) of the CPWM signal (Figure 17-16). If ELSnA is cleared, a compare occurring while counting up negates the CPWM output signal and a compare occurring while counting down asserts the output. The counter counts up until it reaches the modulo setting in TPMxMODH:TPMxMODL, then counts down until it reaches zero. This sets the period equal to two times TPMxMODH:TPMxMODL.
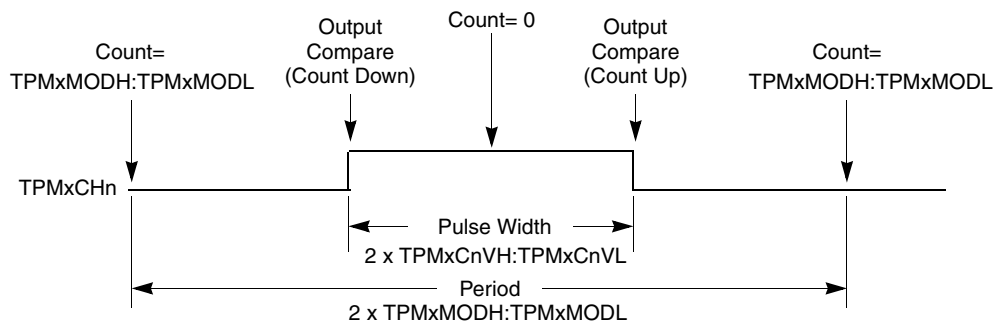
Figure 17-16. CPWM Period and Pulse Width (ELSnA=0)

Center-aligned PWM outputs typically produce less noise than edge-aligned PWMs because fewer I/O pin transitions are lined up at the same system clock edge. This type of PWM is also required for some types of motor drives.

MCF51QE128 MCU Series Reference Manual, Rev. 3

Input capture, output compare, and edge-aligned PWM functions do not make sense when the counter is operating in up/down counting mode so this implies that all active channels within a TPM must be used in CPWM mode when CPWMS is set.

Because the TPM is connected to an 8-bit peripheral bus, the settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to TPMxMODH, TPMxMODL, TPMxCnVH, and TPMxCnVL, actually write to buffer registers.

In center-aligned PWM mode, the TPMxCnVH:L registers are updated with the value of their write buffer according to the value of CLKSB:CLKSA bits:

- If CLKSB and CLKSA are cleared, the registers are updated when the second byte is written
- If CLKSB and CLKSA are not cleared, the registers are updated after both bytes are written, and the TPM counter changes from TPMxMODH:TPMxMODL − 1 to TPMxMODH:TPMxMODL. If the TPM counter is a free-running counter, the update is made when the TPM counter changes from 0xFFFE to 0xFFFF.

When TPMxCNTH:TPMxCNTL equals TPMxMODH:TPMxMODL, the TPM optionally generates a TOF interrupt at the end of this count.

Writing to TPMxSC cancels any values written to TPMxMODH and/or TPMxMODL and resets the coherency mechanism for the modulo registers. Writing to TPMxCnSC cancels any values written to the channel value registers and resets the coherency mechanism for TPMxCnVH:TPMxCnVL.

17.5   Reset Overview

17.5.1   General

The TPM is reset whenever any MCU reset occurs.

17.5.2   Description of Reset Operation

Reset clears TPMxSC, which disables clocks to the TPM and disables timer overflow interrupts (TOIE = 0). CPWMS, MSnB, MSnA, ELSnB, and ELSnA are all cleared. This configures all TPM channels for input-capture operation with the associated pins disconnected from I/O pin logic (all MCU pins related to the TPM revert to general purpose I/O pins).

17.6   Interrupts

17.6.1   General

The TPM generates an optional interrupt for the main counter overflow and an interrupt for each channel. The meaning of channel interrupts depends on each channel's mode of operation. If the channel is configured for input capture, the interrupt flag is set each time the selected input capture edge is recognized. If the channel is configured for output compare or PWM modes, the interrupt flag is set each time the main timer counter matches the value in the 16-bit channel value register.

All TPM interrupts are listed in Table 17-8, showing the interrupt name, the name of any local enable that can block the interrupt request from leaving the TPM and getting recognized by the separate interrupt processing logic.

Table 17-8. Interrupt Summary

Interrupt	Local Enable	Source	Description
TOF	TOIE	Counter overflow	Set each time the timer counter reaches its terminal count (at transition to next count value which is usually 0x0000)
CHnF	CHnIE	Channel event	An input capture or output compare event took place on channel n

The TPM module provides a high-true interrupt signal. Vectors and priorities are determined at chip integration time in the interrupt module. Refer to the interrupt chapter for details.

17.6.2   Description of Interrupt Operation

For each interrupt source in the TPM, a flag bit is set upon recognition of the interrupt condition such as timer overflow, channel-input capture, or output-compare events. This flag may be read (polled) by software to determine that the action has occurred, or an associated enable bit (TOIE or CHnIE) can be set to enable hardware interrupt generation. While the interrupt enable bit is set, a static interrupt generates when the associated interrupt flag is set. Software must perform a sequence of steps to clear the interrupt flag before returning from the interrupt service routine.

TPM interrupt flags are cleared by a two-step process including a read of the flag bit while it is set followed by a write of zero to the bit. If a new event is detected between these two steps, the sequence is reset and the interrupt flag remains set after the second step to avoid the possibility of missing the new event.

17.6.2.1   Timer Overflow Interrupt (TOF) Description

The meaning and details of operation for TOF interrupts varies slightly depending upon the mode of operation of the TPM system (general purpose timing functions versus center-aligned PWM operation). The flag is cleared by the two step sequence described above.

17.6.2.1.1   Normal Case

Normally, TOF is set when the timer counter changes from 0xFFFF to 0x0000. When the TPM is not configured for center-aligned PWM (CPWMS = 0), TOF is set when the timer counter changes from the terminal count (the value in the modulo register) to 0x0000. This case corresponds to the normal meaning of counter overflow.

17.6.2.1.2   Center-Aligned PWM Case

When CPWMS is set, TOF is set when the timer counter changes direction from up-counting to down-counting at the end of the terminal count (the value in the modulo register). In this case, TOF corresponds to the end of a PWM period.

17.6.2.2   Channel Event Interrupt Description

The meaning of channel interrupts depends on the channel's current mode (input-capture, output-compare, edge-aligned PWM, or center-aligned PWM).

17.6.2.2.1   Input Capture Events

When a channel is configured as an input capture channel, the ELSnB and ELSnA control bits select no edge (off), rising edges, falling edges, or any edge as the edge that triggers an input capture event. When the selected edge is detected, the interrupt flag is set. The flag is cleared by the two-step sequence described in Section 17.6.2, "Description of Interrupt Operation."

17.6.2.2.2   Output Compare Events

When a channel is configured as an output compare channel, the interrupt flag is set each time the main timer counter matches the 16-bit value in the channel value register. The flag is cleared by the two-step sequence described Section 17.6.2, "Description of Interrupt Operation."

17.6.2.2.3   PWM End-of-Duty-Cycle Events

There are two possibilities for channels configured for PWM operation. When the channel is configured for edge-aligned PWM, the channel flag is set when the timer counter matches the channel value register that marks the end of the active duty cycle period. When the channel is configured for center-aligned PWM, the timer count matches the channel value register twice during each PWM cycle. In this CPWM case, the channel flag is set at the start and at the end of the active duty cycle period, when the timer counter matches the channel value register. The flag is cleared by the two-step sequence described Section 17.6.2, "Description of Interrupt Operation."

Chapter 18
Version 1 ColdFire Debug (CF1_DEBUG)

18.1 Introduction

This chapter describes the capabilities defined by the Version 1 ColdFire debug architecture. The Version 1 ColdFire core supports BDM functionality using the HCS08's single-pin interface. The traditional 3-pin full-duplex ColdFire BDM serial communication protocol based on 17-bit data packets is replaced with the HCS08 protocol where all communications are based on an 8-bit data packet using a single package pin (BKGD).

An on-chip trace buffer allows a stream of compressed processor execution status packets to be recorded for subsequent retrieval to provide program (and partial data) trace capabilities.

The following sections in this chapter provide details on the BKGD pin, the background debug serial interface controller (BDC), a standard 6-pin BDM connector, the BDM command set as well as real-time debug and trace capabilities. The V1 definition supports revision B+ (DEBUG_B+) of the ColdFire debug architecture.

A simplified block diagram of the V1 core including the processor and debug module is shown in Figure 18-1.

IFP — Instruction fetch pipeline
OEP — Operand execution pipeline
BDC — Background debug controller
CFxBDM — ColdFire background debug module
PST/DDATA — Processor status/debug data
RTD — Real-time debug
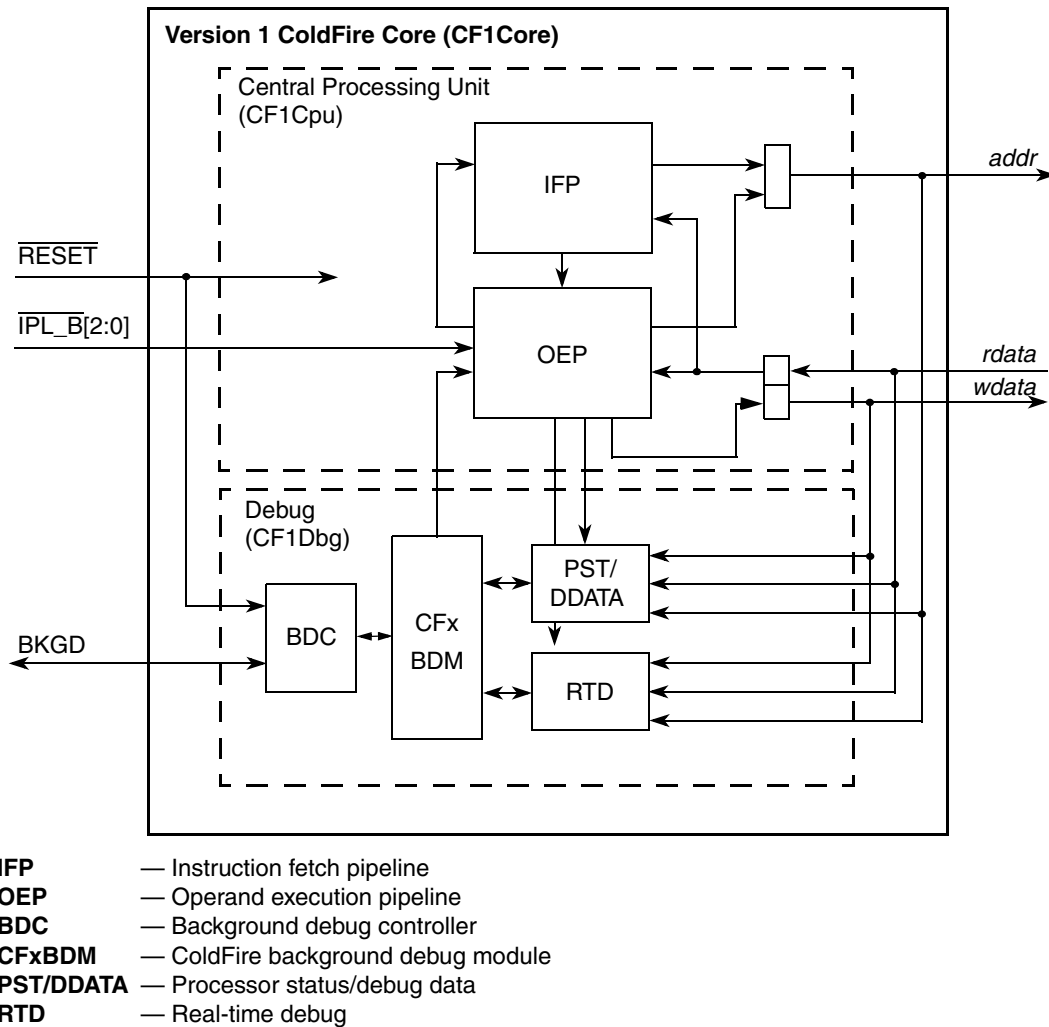
Figure 18-1. Simplified Version 1 ColdFire Core Block Diagram

18.1.1   Overview

Debug support is divided into three areas:

- Background debug mode (BDM)—Provides low-level debugging in the ColdFire processor core. In BDM, the processor core is halted and a variety of commands can be sent to the processor to access memory, registers, and peripherals. The external emulator uses a one-pin serial communication protocol. See Section 18.4.1, "Background Debug Mode (BDM)".

- Real-time debug support—Use of the full BDM command set requires the processor to be halted, which many real-time embedded applications cannot support. The core includes a variety of internal breakpoint registers which can be configured to trigger and generate a special interrupt. The resulting debug interrupts let real-time systems execute a unique service routine that can quickly save the contents of key registers and variables and return the system to normal operation. The external development system can then access the saved data, because the hardware supports

concurrent operation of the processor and BDM-initiated memory commands. In addition, the option is provided to allow interrupts to occur. See Section 18.4.2, "Real-Time Debug Support".

- Program trace support—The ability to determine the dynamic execution path through an application is fundamental for debugging. The V1 solution implements a trace buffer that records processor execution status and data, which can be subsequently accessed by the external emulator system to provide program (and optional partial data) trace information. See Section 18.4.3, "Real-Time Trace Support".

There are two fields in debug registers which provide revision information: the hardware revision level in CSR and the 1-pin debug hardware revision level in CSR2. Table 18-1 summarizes the various debug revisions.

Table 18-1. Debug Revision Summary

Revision	CSR[HRL]	CSR2[D1HRL][1]	Enhancements
A	0000	N/A	Initial ColdFire debug definition
B	0001	N/A	BDM command execution does not affect hardware breakpoint logic Added BDM address attribute register (BAAR) BKPT configurable interrupt (CSR[BKD]) Level 1 and level 2 triggers on OR condition, in addition to AND SYNC_PC command to display the processor's current PC
B+	1001	N/A	3 new PC breakpoint registers PBR1–3
CF1_B+	1001	0000	Converted to HCS08 1-pin BDM serial interface Added PST compression and on-chip PST/DDATA buffer for program trace

[1] CSR2 is only available in Version 1 ColdFire devices.

18.1.2   Features

The Version 1 ColdFire debug definition supports the following features:

- Classic ColdFire DEBUG_B+ functionality mapped into the single-pin BDM interface
- Real time debug support, with 6 hardware breakpoints (4 PC, 1 address and 1 data) that can be configured into a 1- or 2-level trigger with a programmable response (processor halt or interrupt)
- Capture of compressed processor status and debug data into on-chip trace buffer provides program (and optional slave bus data) trace capabilities
- On-chip trace buffer provides programmable start/stop recording conditions plus support for continuous or PC-profiling modes
- Debug resources are accessible via single-pin BDM interface or the privileged WDEBUG instruction from the core

18.1.3   Modes of Operations

V1 ColdFire devices typically implement a number of modes of operation, including run, wait, and stop modes. Additionally, the operation of the core's debug module is highly dependent on a number of chip configurations which determine its operating state.

When operating in secure mode, as defined by a 2-bit field in the flash memory examined at reset, BDM access to debug resources is extremely restricted. It is possible to tell that the device has been secured, and to clear security, which involves mass erasing the on-chip flash memory. No other debug access is allowed. Secure mode can be used in conjunction with each of the wait and stop low-power modes.

If the BDM interface is not enabled, access to the debug resources is limited in the same manner as a secure device.

If the device is not secure and the BDM interface is enabled (XCSR[ENBDM] is set), the device is operating in debug mode and additional resources are available via the BDM interface. In this mode, the mode of the processor (running, stopped, or halted) determines which BDM commands may be used.

Debug mode functions are managed through the background debug controller (BDC) in the Version 1 ColdFire core. The BDC provides the means for analyzing MCU operation during software development.

BDM commands can be classified into three types as shown in Table 18-2.

Table 18-2. BDM Command Types

Command Type	Flash Secure?	BDM?	Core Status	Command Set
Always-available	Secure or Unsecure	Enabled or Disabled	—	• Read/write access to XCSR[31–24], CSR2[31–24], CSR3[31–24]
Non-intrusive	Unsecure	Enabled	Run, Halt	• Memory access • Memory access with status • Debug register access • BACKGROUND
Active background	Unsecure	Enabled	Halt	• Read or write CPU registers (also available in stop mode) • Single-step the application • Exit halt mode to return to the application program (GO)

For more information on these three BDM command classifications, see Section 18.4.1.5, "BDM Command Set Summary."

The core's halt mode is entered in a number of ways:

- The BKGD pin is low during POR.
- The BKGD pin is low immediately after a BDM-initiated force reset (see Section 18.3.3, "Configuration/Status Register 2 (CSR2)," for details).
- A background debug force reset occurs (CSR2[BDFR] is set) and CSR2[BFHBR] is set.
- A BACKGROUND command is received through the BKGD pin. If necessary, this wakes the device from STOP/WAIT modes.
- A properly-enabled (XCSR[ENBDM] is set) HALT instruction is executed.
- Encountering a BDM breakpoint and the trigger response is programmed to generate a halt

While in halt mode, the core waits for serial background commands rather than executing instructions from the application program.
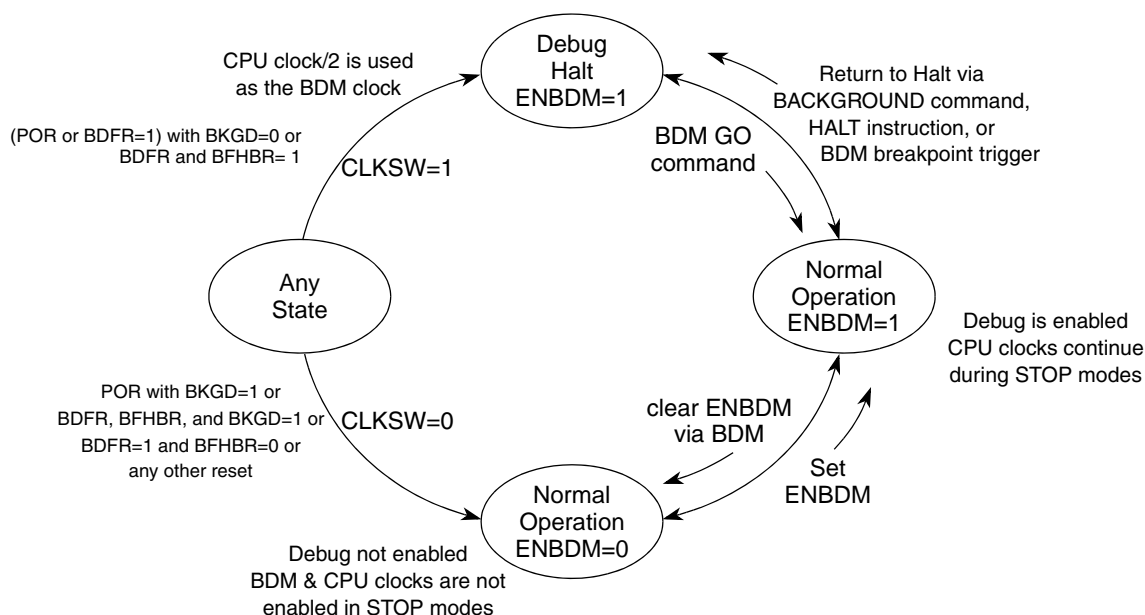
Figure 18-2. Debug Modes State Transition Diagram

Figure 18-2 contains a simplified view of the V1 ColdFire debug mode states. The XCSR[CLKSW] bit controls the BDC clock source. When CLKSW is set, the BDC serial clock is half the CPU clock. When CLKSW is cleared, the BDC serial clock is ICSLCLK. This signal is supplied from the on-chip 10 MHz bus clock/20 MHz CPU clock frequency locked loop. Normally, ICSLCLK should be used when the device in in any of the following clock modes: FBE, FEI, FBI, and FEE. The FLL clocks are not available when the device is in FBELP and FBILP modes. In these cases, ICSOUT should be used.

The ENBDM bit determines if the device can be placed in halt mode, if the core and BDC serial clocks continue to run in STOP modes, and if the regulator can be placed into standby mode. Again, if booting to halt mode, XCSR[ENBDM, CLKSW] are automatically set.

If ENBDM is cleared, the ColdFire core treats the HALT instruction as an illegal instruction and generates a reset (if CPUCR[IRD] is cleared) or an exception (if CPUCR[IRD] is set) if execution is attempted.

If XCSR[ENBDM] is set, the device can be restarted from STOP/WAIT via the BDM interface.

18.2   External Signal Descriptions

Table 18-3 describes the debug module's 1-pin external signal. A standard 6-pin debug connector is shown in Section 18.4.4, "Freescale-Recommended BDM Pinout".

Table 18-3. Debug Module Signals

Signal	Description
Background Debug (BKGD)	Single-wire background debug interface pin. The primary function of this pin is for bidirectional serial communication of background debug mode commands and data. During reset, this pin selects between starting in active background (halt) mode or starting the application program. This pin also requests a timed sync response pulse to allow a host development tool to determine the correct clock frequency for background debug serial communications.

18.3   Memory Map/Register Definition

In addition to the BDM commands that provide access to the processor's registers and the memory subsystem, the debug module contains a number of registers. Most of these registers (all except the PST/DDATA trace buffer) are also accessible (write-only) from the processor's supervisor programming model by executing the WDEBUG instruction. Thus, the breakpoint hardware in the debug module can be read (certain registers) or written by the external development system using the serial debug interface or written by the operating system running on the processor core. Software is responsible for guaranteeing that accesses to these resources are serialized and logically consistent. The hardware provides a locking mechanism in the CSR to allow the external development system to disable any attempted writes by the processor to the breakpoint registers (setting CSR[IPW]). BDM commands must not be issued during the processor's execution of the WDEBUG instruction to configure debug module registers or the resulting behavior is undefined.

These registers, shown in Table 18-4, are treated as 32-bit quantities regardless of the number of implemented bits and unimplemented bits are reserved and must be cleared. These registers are also accessed through the BDM port by the commands, WRITE_DREG and READ_DREG, described in Section 18.4.1.5, "BDM Command Set Summary." These commands contain a 5-bit field, DRc, that specifies the register, as shown in Table 18-4.

Table 18-4. Debug Module Memory Map

DRc	Register Name	Width (bits)	Access	Reset Value	Section/ Page
0x00	Configuration/status register (CSR)	32	R/W (BDM), W (CPU)	0x0090_0000	18.3.1/18-361
0x01	Extended Configuration/Status Register (XCSR)	32	R/W[1] (BDM), W (CPU)	0x0000_0000	18.3.2/18-364
0x02	Configuration/Status Register 2 (CSR2)	32	R/W[1] (BDM), W (CPU)	See Section	18.3.3/18-367
0x03	Configuration/Status Register 3 (CSR3)	32[2]	R/W[1] (BDM), W (CPU)	0x0000_0000	18.3.4/18-370
0x05	BDM address attribute register (BAAR)	32[2]	W	0x0000_0005	18.3.5/18-371
0x06	Address attribute trigger register (AATR)	32[2]	W	0x0000_0005	18.3.6/18-372
0x07	Trigger definition register (TDR)	32	W	0x0000_0000	18.3.7/18-373
0x08	PC breakpoint register 0 (PBR0)	32	W	Undefined, Unaffected	18.3.8/18-376
0x09	PC breakpoint mask register (PBMR)	32	W	Undefined, Unaffected	18.3.8/18-376
0x0C	Address breakpoint high register (ABHR)	32	W	Undefined, Unaffected	18.3.9/18-378
0x0D	Address breakpoint low register (ABLR)	32	W	0x0000_0000	18.3.9/18-378
0x0E	Data breakpoint register (DBR)	32	W	0x0000_0000	18.3.10/18-379
0x0F	Data breakpoint mask register (DBMR)	32	W	0x0000_0000	18.3.10/18-379

Table 18-4. Debug Module Memory Map (continued)

DRc	Register Name	Width (bits)	Access	Reset Value	Section/ Page
0x18	PC breakpoint register 1 (PBR1)	32	W	PBR1[0] = 0	18.3.8/18-376
0x1A	PC breakpoint register 2 (PBR2)	32	W	PBR2[0] = 0	18.3.8/18-376
0x1B	PC breakpoint register 3 (PBR3)	32	W	PBR3[0] = 0	18.3.8/18-376
0x20 + n; n = 0–11	PST Trace Buffer n (PSTBn); n = 0–11	32	R (BDM)[3]	Undefined, Unaffected	18.4.3.2/18-413

[1] The most significant byte of the XCSR, CSR2 and CSR3 registers supports special control functions and are writeable via BDM using the WRITE_XCSR_BYTE, WRITE_CSR2_BYTE, and WRITE_CXSR3_BYTE commands. They can be read from BDM using the READ_XCSR_BYTE, READ_CSR2_BYTE, and READ_CSR3_BYTE commands. These 3 registers, along with the CSR, can also be referenced as 32-bit quantities using the BDM READ_DREG and WRITE_DREG commands.

[2] Each debug register is accessed as a 32-bit value; undefined fields are reserved and must be cleared.

[3] The contents of the PST trace buffer is read from BDM (32 bits per access) using READ_PSTB commands.

NOTE

Debug control registers can be written by the external development system or the CPU through the WDEBUG instruction. These control registers are write-only from the programming model and they can be written through the BDM port using the WRITE_DREG command. In addition, the four configuration/status registers (CSR, XCSR, CSR2, CSR3) can be read through the BDM port using the READ_DREG command.

The ColdFire debug architecture supports a number of hardware breakpoint registers that can be configured into single- or double-level triggers based on the PC or operand address ranges with an optional inclusion of specific data values. The triggers can be configured to halt the processor or generate a debug interrupt exception. Additionally, these same breakpoint registers can be used to specify start/stop conditions for recording in the PST trace buffer.

The core includes four PC breakpoint triggers and a set of operand address breakpoint triggers with two independent address registers (to allow specification of a range) and an optional data breakpoint with masking capabilities. Core breakpoint triggers are accessible through the serial BDM interface or written through the supervisor programming model using the WDEBUG instruction.

18.3.1 Configuration/Status Register (CSR)

CSR defines the debug configuration for the processor and memory subsystem and contains status information from the breakpoint logic. CSR is accessible from the programming model using the WDEBUG instruction and through the BDM port using the READ_DREG and WRITE_DREG commands.

DRc[4:0]: 0x00 (CSR)
Access: Supervisor write-only
BDM read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		BS	TAT		FOF	TRG	HALT	BKPT		HRL			0	BKD	0	IPW
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	TRC	0	DDC		UHE	BTB		0	NPL	IPI	SSM	0	0	FID	DDH
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
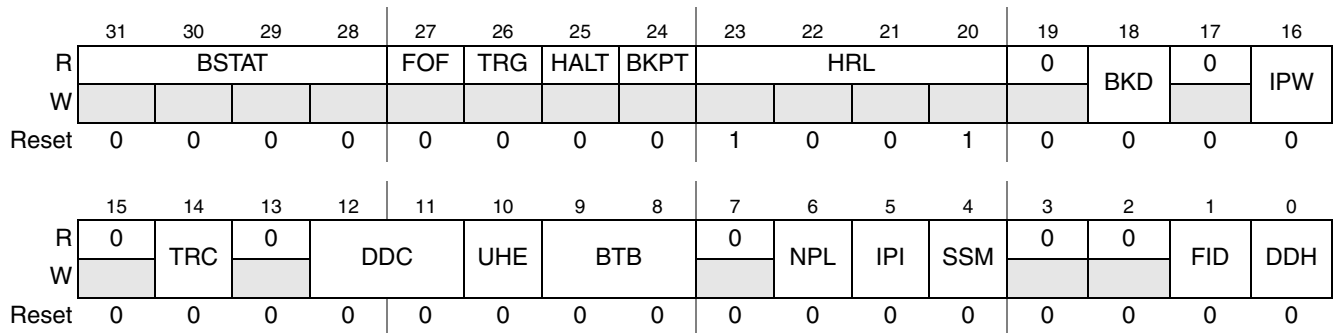
Figure 18-3. Configuration/Status Register (CSR)

Table 18-5. CSR Field Descriptions

Field	Description
31–28 BSTAT	Breakpoint status. Provides read-only status (from the BDM port only) information concerning hardware breakpoints. BSTAT is cleared by a TDR write or by a CSR read when either a level-2 breakpoint is triggered or a level-1 breakpoint is triggered and the level-2 breakpoint is disabled. 0000 No breakpoints enabled 0001 Waiting for level-1 breakpoint 0010 Level-1 breakpoint triggered 0101 Waiting for level-2 breakpoint 0110 Level-2 breakpoint triggered
27 FOF	Fault-on-fault. Indicates a catastrophic halt occurred and forced entry into BDM. FOF is cleared by reset or when CSR is read (from the BDM port only).
26 TRG	Hardware breakpoint trigger. Indicates a hardware breakpoint halted the processor core and forced entry into BDM. Reset, the debug GO command, or reading CSR (from the BDM port only) clears TRG.
25 HALT	Processor halt. Indicates the processor executed a HALT and forced entry into BDM. Reset, the debug GO command, or reading CSR (from the BDM port only) clears HALT.
24 BKPT	Breakpoint assert. Indicates the $\overline{\text{BKPT}}$ input was asserted or a BDM BACKGROUND command received, forcing the processor into a BDM halt. Reset, the debug GO command, or reading CSR (from the BDM port only) clears BKPT.
23–20 HRL	Hardware revision level. Indicates, from the BDM port only, the level of debug module functionality. An emulator can use this information to identify the level of functionality supported. 0000 Revision A 0001 Revision B 0010 Revision C 0011 Revision D 1001 Revision B+ (The value used for this device) 1011 Revision D+
19	Reserved, must be cleared.

Table 18-5. CSR Field Descriptions (continued)

Field	Description
18 BKD	Breakpoint disable. Disables the normal $\overline{BKPT}$ input signal and BACKGROUND command functionality, and allows the assertion of this pin (or execution of the BACKGROUND command) to generate a debug interrupt. 0 Normal operation 1 $\overline{BKPT}$ is edge-sensitive: a high-to-low edge on $\overline{BKPT}$ or the receipt of a BDM BACKGROUND command signals a debug interrupt to the ColdFire core. The processor makes this interrupt request pending until the next sample point occurs, when the exception is initiated. In the ColdFire architecture, the interrupt sample point occurs once per instruction. There is no support for nesting debug interrupts.
17	Reserved, must be cleared.
16 IPW	Inhibit processor writes. Inhibits processor-initiated writes to the debug module's programming model registers. IPW can be modified only by commands from the BDM interface.
15	Reserved, must be cleared.
14 TRC	Force emulation mode on trace exception. 0 Processor enters supervisor mode. 1 Processor enters emulator mode when a trace exception occurs.
13	Reserved, must be cleared.
12–11 DDC	Debug data control. Controls peripheral bus operand data capture for DDATA, which displays the number of bytes defined by the operand reference size (a marker) before the actual data; byte displays 8 bits, word displays 16 bits, and long displays 32 bits (one nibble at a time across multiple PSTCLK clock cycles). See Table 18-26. A non-zero value enables partial data trace capabilities. 00 No operand data is displayed. 01 Capture all write data. 10 Capture all read data. 11 Capture all read and write data.
10 UHE	User halt enable. Selects the CPU privilege level required to execute the HALT instruction. The core must be operating with XCSR[ENBDM] set to execute any HALT instruction, else the instruction is treated as an illegal opcode. 0 HALT is a supervisor-only instruction. 1 HALT is a supervisor/user instruction.
9–8 BTB	Branch target bytes. Defines the number of bytes of branch target address DDATA displays. See Section 18.4.3.1, "Begin Execution of Taken Branch (PST = 0x05)." 00 No target address capture 01 Lower 2 bytes of the target address 1x Lower 3 bytes of the target address
7	Reserved, must be cleared.
6 NPL	Non-pipelined mode. Determines if the core operates in pipelined mode. 0 Pipelined mode 1 Non-pipelined mode. The processor effectively executes one instruction at a time with no overlap. This typically adds five cycles to the execution time of each instruction. Given an average execution latency of ~2 cycles per instruction, throughput in non-pipeline mode would be ~7 cycles per instruction, approximately 25% - 33% of pipelined performance. Regardless of the NPL state, a triggered PC breakpoint is always reported before the triggering instruction executes. In normal pipeline operation, the occurrence of an address and/or data breakpoint trigger is imprecise. In non-pipeline mode, these triggers are always reported before the next instruction begins execution and trigger reporting can be considered precise.

Table 18-5. CSR Field Descriptions (continued)

Field	Description
5 IPI	Ignore pending interrupts when in single-step mode. 0 Core services any pending interrupt requests signalled while in single-step mode. 1 Core ignores any pending interrupt requests signalled while in single-step mode.
4 SSM	Single-step mode enable. 0 Normal mode. 1 Single-step mode. The processor halts after execution of each instruction. While halted, any BDM command can be executed. On receipt of the GO command, the processor executes the next instruction and halts again. This process continues until SSM is cleared.
3–2	Reserved, must be cleared.
1 FID	Force *ipg_debug*. The core generates this output to the device, signaling it is in debug mode. 0 Do not force the assertion of *ipg_debug* 1 Force the assertion of *ipg_debug*
0 DDH	Disable *ipg_debug* due to a halt condition. The core generates an output to the other modules in the device, signaling it is in debug mode. By default, this output signal is asserted whenever the core halts. 0 Assert *ipg_debug* if the core is halted 1 Negate *ipg_debug* due to the core being halted

18.3.2   Extended Configuration/Status Register (XCSR)

The 32-bit XCSR is partitioned into two sections: the upper byte contains status and command bits always accessible to the BDM interface, even if debug mode is disabled. This status byte is also known as XCSR_SB. The lower 24 bits contain fields related to the generation of automatic SYNC_PC commands, which can be used to periodically capture and display the current program counter (PC) in the PST trace buffer (if properly configured).

There are multiple ways to reference the XCSR. They are summarized in Table 18-6.

Table 18-6. XCSR Reference Summary

Method	Reference Details
READ_XCSR_BYTE	Reads XCSR[31−24] from the BDM interface. Available in all modes.
WRITE_XCSR_BYTE	Writes XCSR[31−24] from the BDM interface. Available in all modes.
READ_DREG	Reads XCSR[31−0] from the BDM interface. Classified as a non-intrusive BDM command.
WRITE_DREG	Writes XCSR[31−0] from the BDM interface. Classified as a non-intrusive BDM command.
WDEBUG instruction	Writes XCSR[23−0] during the core's execution of WDEBUG instruction. This instruction is a privileged supervisor-mode instruction.

DRc: 0x01 (XCSR)

Access: Supervisor write-only
BDM read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CPU HALT	CPU STOP	CSTAT			CLKSW	SEC	ENBDM	0	0	0	0	0	0	0	0
W			ESEQC				ERASE									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	APCSC		APCE NB
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-4. Extended Configuration/Status Register (XCSR)

Table 18-7. XCSR Field Descriptions

Field	Description
31 CPUHALT	Indicates that the CPU is in the halt state. The CPU state may be running, stopped, or halted, which is determined by the CPUHALT and CPUSTOP bits as shown below. | XCSR [CPUHALT]	XCSR [CPUSTOP]	CPU State
---	---	---
0	0	Running
0	1	Stopped
1	0	Halted	
30 CPUSTOP	Indicates that the CPU is in the stop state. The CPU state may be running, stopped, or halted, which is determined by the CPUHALT and CPUSTOP bits as shown in the CPUHALT bit description.

Table 18-7. XCSR Field Descriptions (continued)

Field	Description

29–27
CSTAT (R)
ESEQC (W)
	During reads, indicates the BDM command status.
	000 Command done, no errors
	001 Command done, data invalid
	01*x* Command done, illegal
	1*xx* Command busy, overrun
	
	If an overrun is detected (CSTAT = 1*xx*), the following sequence is suggested to clear the source of the error:
	1. Issue a SYNC command to reset the BDC channel.
	2. The host issues a BDM NOP command.
	3. The host checks the channel status using a READ_XCSR_BYTE command.
	4. If XCSR[CSTAT] = 000
	    then status is okay; proceed
	    else
	        Halt the CPU with a BDM BACKGROUND command
	        Repeat steps 1,2,3
	        If XCSR[CSTAT] ≠ 000, then reset device
	
	During writes, the ESEQC field is used for the erase sequence control during flash programming. ERASE must also be set for this bit to have an effect.
	000 User mass erase
	Else Reserved
	Note: See Figure 4-15 for a detailed description of the algorithm for clearing of security.

26 CLKSW	Select source for serial BDC communication clock. 0 Alternate, asynchronous BDC clock, typically 10 MHz 1 Synchronous bus clock (CPU clock divided by 2) The initial state of the XCSR[CLKSW] bit is loaded by the hardware in response to certain reset events and the state of the BKGD pin as described in Figure 18-2.
25 SEC (R) ERASE (W)	The read value of this bit typically defines the status of the flash security field: 0 Flash security is disabled 1 Flash security is enabled In addition, the SEC bit is context-sensitive during reads. Once a mass-erase sequence has been initiated by BDM, it acts as a flash busy flag. When the erase operation is complete and the bit is cleared, it returns to reflect the status of the chip security. 0 Flash is not busy performing a BDM mass-erase sequence 1 Flash is busy performing a BDM mass-erase sequence During writes, this bit qualifies XCSR[ESEQC] for the write modes shown in the ESEQC field description. 0 Do not perform a mass-erase of the flash. 1 Perform a mass-erase of the flash, using the sequence specified in the XCSR[ESEQC] field.
24 ENBDM	Enable BDM. 0 BDM mode is disabled 1 Active background mode is enabled (assuming the flash is not secure)
23–3	Reserved for future use by the debug module, must be cleared.

Table 18-7. XCSR Field Descriptions (continued)

Field	Description
2–1 APCSC	Automatic PC synchronization control. Determines the periodic interval of PC address captures, if XCSR[APCENB] is set. When the selected interval is reached, a SYNC_PC command is sent to the ColdFire CPU. For more information on the SYNC_PC operation, see the APCENB description. The chosen frequency depends on CSR2[APCDIV16] as shown in the equation and table below: $$\text{PC address capture period} = \frac{2^{(APCSC + 1)} \times 1024}{16^{APCDIV16}}$$ *Eqn. 18-1* {{TABLE_18_7_INNER}}
0 APCENB	Automatic PC synchronization enable. Enables the periodic output of the PC which can be used for PST/DDATA trace synchronization. As described in XCSR[APCSC], when the enabled periodic timer expires, a SYNC_PC command is sent to the ColdFire CPU which generates a forced instruction fetch of the next instruction. The PST/DDATA module captures the target address as defined by CSR[9] (two bytes if CSR[9] is cleared, three bytes if CSR[9] is set). This produces a PST sequence of 0x5 (indicating a taken branch), followed by the PST marker indicating a 2- or 3-byte address, and then the captured instruction address. 0 Automatic PC synchronization disabled 1 Automatic PC synchronization enabled

Inner table referenced above:

XCSR [APCENB]	CSR2 [APCDIV16]	XCSR [APCSC]	SYNC_PC Interval
1	0	00	2048 cycles
1	0	01	4096 cycles
1	0	10	8192 cycles
1	0	11	16384 cycles
1	1	00	128 cycles
1	1	01	256 cycles
1	1	10	512 cycles
1	1	11	1024 cycles

18.3.3 Configuration/Status Register 2 (CSR2)

The 32-bit CSR2 is partitioned into two sections. The upper byte contains status and configuration bits always accessible to the BDM interface, even if debug mode is disabled. The lower 24 bits contain fields related to the configuration of the PST trace buffer (PSTB).

There are multiple ways to reference CSR2. They are summarized in Table 18-8.

Table 18-8. CSR2 Reference Summary

Method	Reference Details
READ_CSR2_BYTE	Reads CSR2[31−24] from the BDM interface. Available in all modes.
WRITE_CSR2_BYTE	Writes CSR2[31−24] from the BDM interface. Available in all modes.
READ_DREG	Reads CSR2[31−0] from the BDM interface. Classified as a non-intrusive BDM command.

Table 18-8. CSR2 Reference Summary (continued)

Method	Reference Details
WRITE_DREG	Writes CSR2[31−0] from the BDM interface. Classified as a non-intrusive BDM command.
WDEBUG Instruction	Writes CSR2[23−0] during the core's execution of WDEBUG instruction. This instruction is a privileged supervisor-mode instruction.

DRc: 0x02 (CSR2)　　　　　　　　　　　　　　　　　　　　　　Access: Supervisor read-only
　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　BDM read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PSTBP	0	0	0	0	0	BFHBR	0	0PSTBH	PSTBST		0	D1HRL			
W								BDFR								
Power-on Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Other Reset	0	0	u	u	u	0	u	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PSTBWA								0	APCD	0	PSTBRM		PSTBSS		
W									PSTBR	IV16						
Reset	Unaffected and Undefined								0	0	0	0	0	0	0	0

Figure 18-5. Configuration/Status Register 2 (CSR2)

Table 18-9. CSR2 Field Descriptions

Field	Description
31 PSTBP	PST buffer stop. Signals if a PST buffer stop condition has been reached. 0 A PST trace buffer stop condition has not been reached 1 A PST trace buffer stop condition has been reached
30–26	Reserved, must be cleared.
25 BFHBR	BDM force halt on BDM reset. Determines operation of the device after a BDM reset. This bit is cleared after a power-on reset and is unaffected by any other reset. 0 The device enters normal operation mode following a BDM reset. 1 The device enters in halt mode following a BDM reset, as if the BKGD pin was held low after a power-on-reset or standard BDM-initiated reset.
24 BDFR	Background debug force reset. Forces a BDM reset to the device. This bit always reads as 0 after the reset has been initiated. 0 No reset initiated. 1 Force a BDM reset.
20	Reserved, must be cleared.
23 PSTBH	PST trace buffer halt. Indicates if the processor is halted due to the PST trace buffer being full when recording in a continuous mode. 0 PST trace buffer not full 1 CPU halted due to PST trace buffer being full in continuous mode

Table 18-9. CSR2 Field Descriptions (continued)

Field	Description
22–21 PSTBST	PST trace buffer state. Indicates the current state of the PST trace buffer recording. 00 PSTB disabled 01 PSTB enabled and waiting for the start condition 10 PSTB enabled, recording and waiting for the stop condition 11 PSTB enabled, completed recording after the stop condition was reached
20	Reserved, must be cleared.
19–16 D1HRL	Debug 1-pin hardware revision level. Indicates the hardware revision level of the 1-pin debug module implemented in the ColdFire core. For this device, this field is 0x0.
15–8 PSTBWA	PST trace buffer write address. Indicates the current write address of the PST trace buffer. The most-significant-bit of this field is sticky, that is, once set, it remains set until a PST/DDATA reset event occurs. As the ColdFire core inserts PST and DDATA packets into the trace buffer, this field is incremented. The value of the write address defines the next location in the PST trace buffer to be loaded. In other words, the contents of PSTB[PSTBWA-1] is the last valid entry in the trace buffer. The msb of this field can be used to determine if the entire PST trace buffer has been loaded with valid data. <table><tr><td>**PSTBWA[7]**</td><td>**PSTB Valid Data Locations (Oldest to Newest)**</td></tr><tr><td>0</td><td>0, 1, ... PSTBWA-1</td></tr><tr><td>1</td><td>PSTBWA, PSTBWA+1,..., 0, 1, PSTBWA-1</td></tr></table> The PSTBWA is unaffected when a buffer stop condition has been reached, the buffer is disabled, or a system reset occurs. This allows the contents of the PST trace buffer to be retrieved after these events to assist in debug.
7 PSTBR	PST trace buffer reset. Generates a reset of the PST trace buffer logic, which clears PSTBWA and PSTBST. The same resources are reset whenever a disabled trace buffer becomes enabled and upon the receipt of a BDM GO command when operating in continuous trace mode. These reset events also clear any accumulation of PSTs. This bit always reads as a zero. 0 Do not force a PST trace buffer reset 1 Force a PST trace buffer reset
6 APCDIV16	Automatic PC synchronization divide cycle counts by 16. This bit divides the cycle counts for automatic SYNC_PC command insertion by 16. See the APCSC and APCENB field descriptions.
5	Reserved, must be cleared.

Table 18-9. CSR2 Field Descriptions (continued)

Field	Description
4–3 PSTBRM	PST trace buffer recording mode. Defines the trace buffer recording mode. The start and stop recording conditions are defined by the PSTBSS field. 00 Normal recording mode 01 Continuous, normal recordingReserved 10 PC profile recordingReserved 11 Continuous PC profile recordingReserved
2–0 PSTBSS	PST trace buffer start/stop definition. Specifies the start and stop conditions for PST trace buffer recording. In certain cases, the start and stop conditions are defined by the breakpoint registers. The remaining breakpoint registers are available for trigger configurations. <table><tr><th>PSTBSS</th><th>Start Condition</th><th>Stop Condition</th></tr><tr><td>000</td><td colspan="2">Trace buffer disabled, no recording</td></tr><tr><td>001</td><td colspan="2">Unconditional recording</td></tr><tr><td>010</td><td rowspan="2">AB*x*R{& DBR/DBMR}</td><td>PBR0/PBMR</td></tr><tr><td>011</td><td>PBR1</td></tr><tr><td>100</td><td rowspan="2">PBR0/PBMR</td><td>AB*x*R{& DBR/DBMR}</td></tr><tr><td>101</td><td>PBR1</td></tr><tr><td>110</td><td rowspan="2">PBR1</td><td>AB*x*R{& DBR/DBMR}</td></tr><tr><td>111</td><td>PBR0/PBMR</td></tr></table>

18.3.4 Configuration/Status Register 3 (CSR3)

CSR3 contains the BDM flash clock divider (BFCDIV) value in a format similar to HCS08 devices.

There are multiple ways to reference CSR3. They are summarized in Table 18-10.

Table 18-10. CSR3 Reference Summary

Method	Reference Details
READ_CSR3_BYTE	Reads CSR3[31−24] from the BDM interface. Available in all modes.
WRITE_CSR3_BYTE	Writes CSR3[31−24] from the BDM interface. Available in all modes.
READ_DREG	Reads CSR3[31−0] from the BDM interface. Classified as a non-intrusive BDM command.
WRITE_DREG	Writes CSR3[31−0] from the BDM interface. Classified as a non-intrusive BDM command.
WDEBUG Instruction	No operation during the core's execution of a WDEBUG instruction

DRc: 0x03 (CSR3) Access: Supervisor write-only
BDM read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	BFC DIV8			BFCDIV				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-6. Configuration/Status Register 3 (CSR3)

Table 18-11. CSR3 Field Descriptions

Field	Description
31	Reserved, must be cleared.
30 BFCDIV8	BDM flash clock divide by 8. 0 Input to the flash clock divider is the bus clock 1 Input to the flash clock divider is the bus clock divided by 8
29–24 BFCDIV	BDM flash clock divider. The BFCDIV8 and BFCDIV fields specify the frequency of the internal flash clock when performing a mass erase operation initiated by setting XCSR[ERASE]. These fields must be loaded with the appropriate values prior to the setting of XCSR[ERASE] to initiate a mass erase operation in the flash memory. This field divides the bus clock (or the bus clock divided by 8 if BFCDIV8 is set) by the value defined by the BFCDIV plus one. The resulting frequency of the internal flash clock must fall within the range of 150–200 kHz for proper flash operations. Program/erase timing pulses are one cycle of this internal flash clock, which corresponds to a range of 5–6.7 µs. The automated programming logic uses an integer number of these pulses to complete an erase or program operation. if BFCDIV8 = 0, then $f_{FCLK} = f_{Bus} \div (BFCDIV + 1)$ if BFCDIV8 = 1, then $f_{FCLK} = f_{Bus} \div (8 \times (BFCDIV + 1))$ where f_{FCLK} is the frequency of the flash clock and f_{Bus} is the frequency of the bus clock.
23–0	Reserved for future use by the debug module, must be cleared.

18.3.5 BDM Address Attribute Register (BAAR)

BAAR defines the address space for memory-referencing BDM commands. BAAR[R, SZ] are loaded directly from the BDM command, while the lower five bits can be programmed from the external development system. BAAR is loaded any time AATR is written and is initialized to a value of 0x05, setting supervisor data as the default address space. The upper 24 bits of this register are reserved for future use and any attempted write of these bits is ignored.

DRc: 0x05 (BAAR) Access: Supervisor write-only
BDM write-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	R	SZ		TT		TM		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

Figure 18-7. BDM Address Attribute Register (BAAR)

Table 18-12. BAAR Field Descriptions

Field	Description
31–8	Reserved for future use by the debug module, must be cleared.
7 R	Read/Write. 0 Write 1 Read
6–5 SZ	Size. 00 Longword 01 Byte 10 Word 11 Reserved
4–3 TT	Transfer type. See the TT definition in the AATR description, Section 18.3.6, "Address Attribute Trigger Register (AATR)".
2–0 TM	Transfer modifier. See the TM definition in the AATR description, Section 18.3.6, "Address Attribute Trigger Register (AATR)".

18.3.6 Address Attribute Trigger Register (AATR)

AATR defines address attributes and a mask to be matched in the trigger. The register value is compared with address attribute signals from the processor's high-speed local bus, as defined by the setting of the trigger definition register (TDR). AATR is accessible in supervisor mode as debug control register 0x06 using the WDEBUG instruction and through the BDM port using the WRITE_DREG command.

DRc: 0x06 (AATR) Access: Supervisor write-only
BDM write-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RM	SZM	TTM		TMM			R	SZ		TT		TM			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
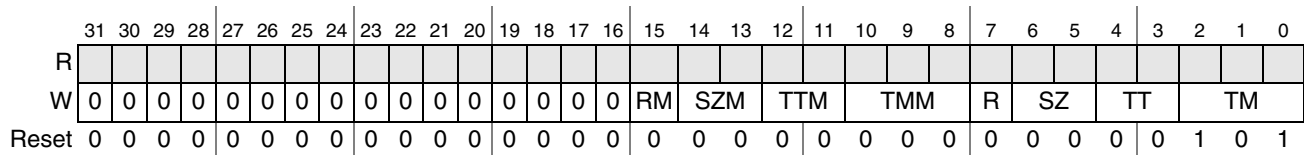
Figure 18-8. Address Attribute Trigger Register (AATR)

Table 18-13. AATR Field Descriptions

Field	Description
31–16	Reserved, must be cleared.
15 RM	Read/write mask. Masks the R bit in address comparisons.

Table 18-13. AATR Field Descriptions (continued)

Field	Description
14–13 SZM	Size mask. Masks the corresponding SZ bit in address comparisons.
12–11 TTM	Transfer type mask. Masks the corresponding TT bit in address comparisons.
10–8 TMM	Transfer modifier mask. Masks the corresponding TM bit in address comparisons.
7 R	Read/write. R is compared with the R/$\overline{W}$ signal of the processor's local bus.
6–5 SZ	Size. Compared to the processor's local bus size signals. 00 Longword 01 Byte 10 Word 11 Reserved
4–3 TT	Transfer type. Compared with the local bus transfer type signals. These bits also define the TT encoding for BDM memory commands. 00 Normal processor access Else Reserved
2–0 TM	Transfer modifier. Compared with the local bus transfer modifier signals, which give supplemental information for each transfer type. These bits also define the TM encoding for BDM memory commands (for backward compatibility). 000 Reserved 001 User-mode data access 010 User-mode code access 011 Reserved 100 Reserved 101 Supervisor-mode data access 110 Supervisor-mode code access 111 Reserved

18.3.7   Trigger Definition Register (TDR)

TDR configures the operation of the hardware breakpoint logic that corresponds with the ABHR/ABLR/AATR, PBR/PBR1/PBR2/PBR3/PBMR, and DBR/DBMR registers within the debug module. TDR controls the actions taken under the defined conditions. Breakpoint logic may be configured as one- or two-level trigger. TDR[31–16] defines the second-level trigger, and TDR[15–0] defines the first-level trigger.

NOTE

The debug module has no hardware interlocks. To prevent spurious breakpoint triggers while the breakpoint registers are being loaded, disable TDR (clear TDR[L2EBL,L1EBL]) before defining triggers.

A write to TDR clears the CSR trigger status bits, CSR[BSTAT]. TDR is accessible in supervisor mode as debug control register 0x07 using the WDEBUG instruction and through the BDM port using the WRITE_DREG command.

DRc: 0x07 (TDR)   Access: Supervisor write-only
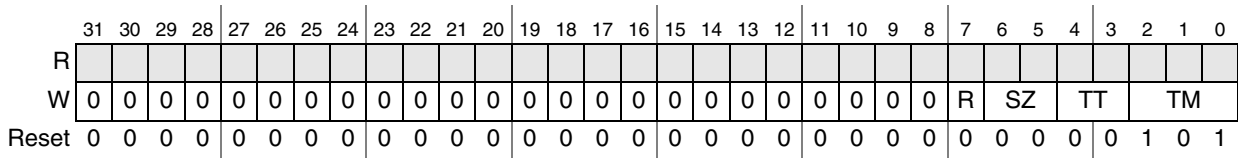   BDM write-only

Second Level Trigger

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	TRC		L2EBL				L2ED				L2DI		L2EA		L2EPC	L2PCI
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

First Level Trigger

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	L2T	L1T	L1EBL				L1ED				L1DI		L1EA		L1EPC	L1PCI
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
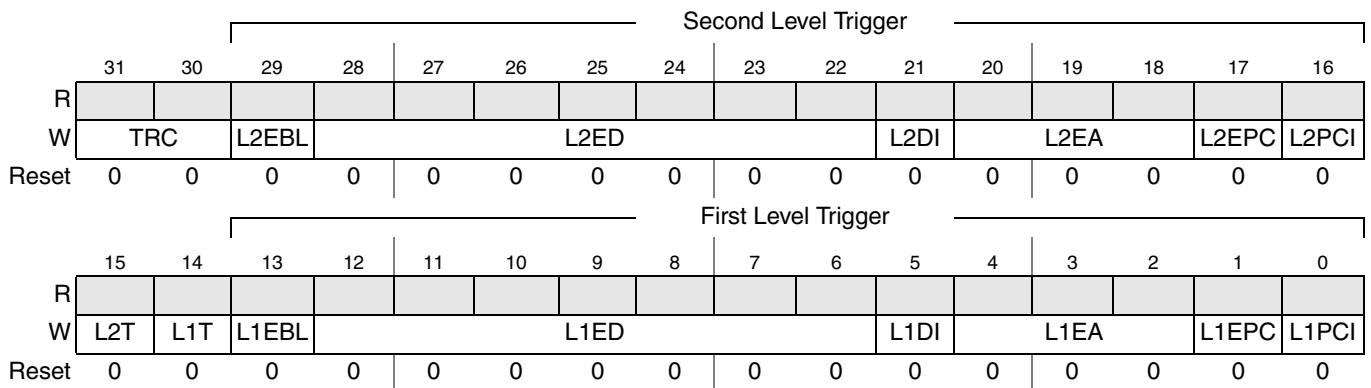
Figure 18-9. Trigger Definition Register (TDR)

Table 18-14. TDR Field Descriptions

Field	Description
31–30 TRC	Trigger response control. Determines how the processor responds to a completed trigger condition. The trigger response is displayed on PST. 00 Display on PST only 01 Processor halt 10 Debug interrupt 11 Reserved
29 L2EBL	Enable level 2 breakpoint. Global enable for the breakpoint trigger. 0 Disables all level 2 breakpoints 1 Enables all level 2 breakpoint triggers
28–22 L2ED	Enable level 2 data breakpoint. Setting an L2ED bit enables the corresponding data breakpoint condition based on the size and placement on the processor's local data bus. Clearing all ED bits disables data breakpoints.<table><tr><th>TDR Bit</th><th>Description</th></tr><tr><td>28</td><td>Data longword. Entire processor's local data bus.</td></tr><tr><td>27</td><td>Lower data word.</td></tr><tr><td>26</td><td>Upper data word.</td></tr><tr><td>25</td><td>Lower lower data byte. Low-order byte of the low-order word.</td></tr><tr><td>24</td><td>Lower middle data byte. High-order byte of the low-order word.</td></tr><tr><td>23</td><td>Upper middle data byte. Low-order byte of the high-order word.</td></tr><tr><td>22</td><td>Upper upper data byte. High-order byte of the high-order word.</td></tr></table>
21 L2DI	Level 2 data breakpoint invert. Inverts the logical sense of all the data breakpoint comparators. This can develop a trigger based on the occurrence of a data value other than the DBR contents. 0 No inversion 1 Invert data breakpoint comparators.

Table 18-14. TDR Field Descriptions (continued)

Field	Description
20–18 L2EA	Enable level 2 address breakpoint. Setting an L2EA bit enables the corresponding address breakpoint. Clearing all three bits disables the breakpoint. <table><tr><th>TDR Bit</th><th>Description</th></tr><tr><td>20</td><td>Address breakpoint inverted. Breakpoint is based outside the range between ABLR and ABHR.</td></tr><tr><td>19</td><td>Address breakpoint range. The breakpoint is based on the inclusive range defined by ABLR and ABHR.</td></tr><tr><td>18</td><td>Address breakpoint low. The breakpoint is based on the address in the ABLR.</td></tr></table>
17 L2EPC	Enable level 2 PC breakpoint. 0 Disable PC breakpoint 1 Enable PC breakpoint
16 L2PCI	Level 2 PC breakpoint invert. 0 The PC breakpoint is defined within the region defined by PBR*n* and PBMR. 1 The PC breakpoint is defined outside the region defined by PBR*n* and PBMR.
15 L2T	Level 2 trigger. Determines the logic operation for the trigger between the PC_condition and the (Address_range and Data) condition where the inclusion of a Data_condition is optional. The ColdFire debug architecture supports the creation of single or double-level triggers. 0 Level 2 trigger = PC_condition & (Address_range & Data_condition) 1 Level 2 trigger = PC_condition | (Address_range & Data_condition)
14 L1T	Level 1 trigger. Determines the logic operation for the trigger between the PC_condition and the (Address_range and Data) condition where the inclusion of a Data_condition is optional. The ColdFire debug architecture supports the creation of single or double-level triggers. 0 Level 1 trigger = PC_condition & (Address_range & Data_condition) 1 Level 1 trigger = PC_condition | (Address_range & Data_condition)
13 L1EBL	Enable level 1 breakpoint. Global enable for the breakpoint trigger. 0 Disables all level 1 breakpoints 1 Enables all level 1 breakpoint triggers
12–6 L1ED	Enable level 1 data breakpoint. Setting an L1ED bit enables the corresponding data breakpoint condition based on the size and placement on the processor's local data bus. Clearing all L1ED bits disables data breakpoints. <table><tr><th>TDR Bit</th><th>Description</th></tr><tr><td>12</td><td>Data longword. Entire processor's local data bus.</td></tr><tr><td>11</td><td>Lower data word.</td></tr><tr><td>10</td><td>Upper data word.</td></tr><tr><td>9</td><td>Lower lower data byte. Low-order byte of the low-order word.</td></tr><tr><td>8</td><td>Lower middle data byte. High-order byte of the low-order word.</td></tr><tr><td>7</td><td>Upper middle data byte. Low-order byte of the high-order word.</td></tr><tr><td>6</td><td>Upper upper data byte. High-order byte of the high-order word.</td></tr></table>

MCF51QE128 MCU Series Reference Manual, Rev. 3

Get the latest version from freescale.com

Table 18-14. TDR Field Descriptions (continued)

Field	Description
5 L1DI	Level 1 data breakpoint invert. Inverts the logical sense of all the data breakpoint comparators. This can develop a trigger based on the occurrence of a data value other than the DBR contents. 0 No inversion 1 Invert data breakpoint comparators.
4–2 L1EA	Enable level 1 address breakpoint. Setting an L1EA bit enables the corresponding address breakpoint. Clearing all three bits disables the address breakpoint. <table><tr><th>TDR Bit</th><th>Description</th></tr><tr><td>4</td><td>Enable address breakpoint inverted. Breakpoint is based outside the range between ABLR and ABHR.</td></tr><tr><td>3</td><td>Enable address breakpoint range. The breakpoint is based on the inclusive range defined by ABLR and ABHR.</td></tr><tr><td>2</td><td>Enable address breakpoint low. The breakpoint is based on the address in the ABLR.</td></tr></table>
1 L1EPC	Enable level 1 PC breakpoint. 0 Disable PC breakpoint 1 Enable PC breakpoint
0 L1PCI	Level 1 PC breakpoint invert. 0 The PC breakpoint is defined within the region defined by PBR*n* and PBMR. 1 The PC breakpoint is defined outside the region defined by PBR*n* and PBMR.

18.3.8 Program Counter Breakpoint/Mask Registers (PBR0–3, PBMR)

The PBR*n* registers define instruction addresses for use as part of the trigger. These registers' contents are compared with the processor's program counter register when the appropriate valid bit is set (for PBR1–3) and TDR is configured appropriately. PBR0 bits are masked by setting corresponding PBMR bits (PBMR has no effect on PBR1–3). Results are compared with the processor's program counter register, as defined in TDR. The PC breakpoint registers, PBR1–3, have no masking associated with them, but do include a valid bit. These registers' contents are compared with the processor's program counter register when TDR is configured appropriately.

The PC breakpoint registers are accessible in supervisor mode using the WDEBUG instruction and through the BDM port using the WRITE_DREG command using values shown in Section 18.4.1.4, "BDM Command Set Descriptions".

NOTE

Version 1 ColdFire core devices implement a 24-bit, 16-Mbyte address map. When programming these registers with a 32-bit address, the upper byte should be zero-filled when referencing the flash, RAM, and RGPIO regions, and set to 0xFF when referencing any of the slave peripheral devices.

DRc: 0x08 (PBR0) Access: Supervisor write-only
BDM write-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																																
W													Address																			
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
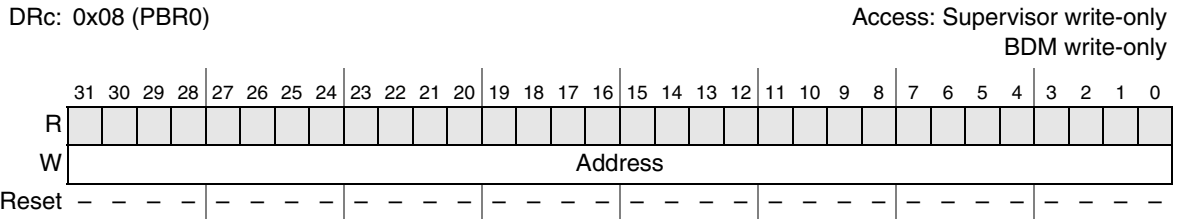
Figure 18-10. Program Counter Breakpoint Register 0 (PBR0)

Table 18-15. PBR0 Field Descriptions

Field	Description
31–0 Address	PC breakpoint address. The address to be compared with the PC as a breakpoint trigger. Since all instruction sizes are multiples of 2 bytes, bit 0 of the address should always be zero.

DRc: 0x18 (PBR1) Access: Supervisor write-only
0x1A (PBR2) BDM write-only
0x1C (PBR3)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																																
W													Address																			V
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	0
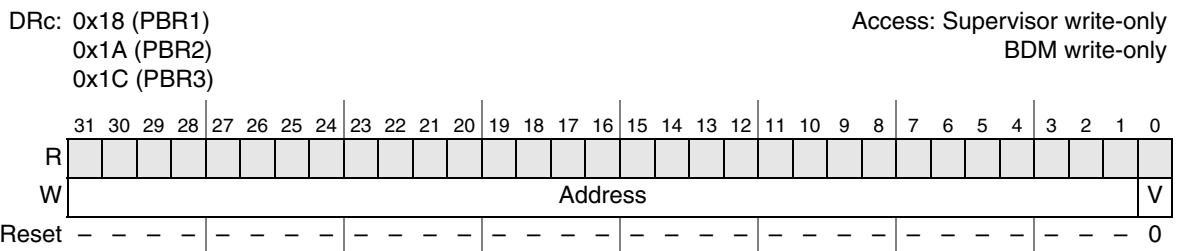
Figure 18-11. Program Counter Breakpoint Register *n* (PBR*n, n = 1,2,3*)

Table 18-16. PBR*n* Field Descriptions

Field	Description
31–1 Address	PC breakpoint address. The 31-bit address to be compared with the PC as a breakpoint trigger.
0 V	Valid bit. This bit must be set for the PC breakpoint to occur at the address specified in the Address field. 0 PBR is disabled. 1 PBR is enabled.

Figure 18-12 shows PBMR. PBMR is accessible in supervisor mode using the WDEBUG instruction and via the BDM port using the WRITE_DREG command. PBMR only masks PBR0.

DRc: 0x09 (PBMR) Access: Supervisor write-only
BDM write-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																																
W													Mask																			
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Figure 18-12. Program Counter Breakpoint Mask Register (PBMR)

MCF51QE128 MCU Series Reference Manual, Rev. 3

Table 18-17. PBMR Field Descriptions

Field	Description
31–0 Mask	PC breakpoint mask. 0 The corresponding PBR0 bit is compared to the appropriate PC bit. 1 The corresponding PBR0 bit is ignored.

18.3.9   Address Breakpoint Registers (ABLR, ABHR)

The ABLR and ABHR define regions in the processor's data address space that can be used as part of the trigger. These register values are compared with the address for each transfer on the processor's high-speed local bus. The trigger definition register (TDR) identifies the trigger as one of three cases:

- Identical to the value in ABLR
- Inside the range bound by ABLR and ABHR inclusive
- Outside that same range

The address breakpoint registers are accessible in supervisor mode using the WDEBUG instruction and through the BDM port using the WRITE_DREG command using values shown in

NOTE

Version 1 ColdFire core devices implement a 24-bit, 16-Mbyte address map. When programming these registers with a 32-bit address, the upper byte should be zero-filled when referencing the flash, RAM, and RGPIO regions, and set to 0xFF when referencing any of the slave peripheral devices.

Figure 18-13. Address Breakpoint Registers (ABLR, ABHR)

Table 18-18. ABLR Field Description

Field	Description
31–0 Address	Low address. Holds the 32-bit address marking the lower bound of the address breakpoint range. Breakpoints for specific addresses are programmed into ABLR.

Table 18-19. ABHR Field Description

Field	Description
31–0 Address	High address. Holds the 32-bit address marking the upper bound of the address breakpoint range.

18.3.10 Data Breakpoint and Mask Registers (DBR, DBMR)

DBR specifies data patterns used as part of the trigger into debug mode. DBR bits are masked by setting corresponding DBMR bits, as defined in TDR.

DBR and DBMR are accessible in supervisor mode using the WDEBUG instruction and through the BDM port using the WRITE_DREG commands.

DRc[4:0]: 0x0E (DBR)
0x0F (DBMR)

Access: Supervisor write-only
BDM write-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																																
W												Data (DBR); Mask (DBMR)																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-14. Data Breakpoint & Mask Registers (DBR, DBMR)

Table 18-20. DBR Field Descriptions

Field	Description
31–0 Data	Data breakpoint value. Contains the value to be compared with the data value from the processor's local bus as a breakpoint trigger.

Table 18-21. DBMR Field Descriptions

Field	Description
31–0 Mask	Data breakpoint mask. The 32-bit mask for the data breakpoint trigger. 0 The corresponding DBR bit is compared to the appropriate bit of the processor's local data bus 1 The corresponding DBR bit is ignored

The DBR supports aligned and misaligned references. Table 18-22 shows the relationships between processor address, access size, and location within the 32-bit data bus.

Table 18-22. Access Size and Operand Data Location

Address[1−0]	Access Size	Operand Location
00	Byte	D[31−24]
01	Byte	D[23−16]
10	Byte	D[15−8]
11	Byte	D[7−0]
0*x*	Word	D[31−16]
1*x*	Word	D[15−0]
xx	Longword	D[31−0]

18.3.11 Resulting Set of Possible Trigger Combinations

The resulting set of possible breakpoint trigger combinations consists of the following options where ||
denotes logical OR, && denotes logical AND, and {} denotes an optional additional trigger term:

One-level triggers of the form:

```
if      (PC_breakpoint)
if      (PC_breakpoint || Address_breakpoint{&& Data_breakpoint})
if      (Address_breakpoint {&& Data_breakpoint})
```

Two-level triggers of the form:

```
if      (PC_breakpoint)
        then if (Address_breakpoint{&& Data_breakpoint})

if      (Address_breakpoint {&& Data_breakpoint})
        then if (PC_breakpoint)
```

In these examples, PC_breakpoint is the logical summation of the PBR0/PBMR, PBR1, PBR2, and PBR3
breakpoint registers; Address_breakpoint is a function of ABHR, ABLR, and AATR; Data_breakpoint is
a function of DBR and DBMR. In all cases, the data breakpoints can be included with an address
breakpoint to further qualify a trigger event as an option.

The breakpoint registers can also be used to define the start and stop recording conditions for the PST trace
buffer. For information on this functionality, see Section 18.3.3, "Configuration/Status Register 2
(CSR2)".

18.4 Functional Description

18.4.1 Background Debug Mode (BDM)

This section provides details on the background debug serial interface controller (BDC) and the BDM
command set.

The BDC provides a single-wire debug interface to the target MCU. As shown in the Version 1 ColdFire core block diagram of Figure 18-1, the BDC module interfaces between the single-pin (BKGD) interface and the remaining debug modules, including the ColdFire background debug logic, the real-time debug hardware, and the PST/DDATA trace logic. This interface provides a convenient means for programming the on-chip flash and other non-volatile memories. The BDC is the primary debug interface for development and allows non-intrusive access to memory data and traditional debug features such as run/halt control, read/write of core registers, breakpoints, and single instruction step.

Features of the background debug controller (BDC) include:

- Single dedicated pin for mode selection and background communications
- Special BDC registers not located in system memory map
- SYNC command to determine target communications rate
- Non-intrusive commands for memory access
- Active background (halt) mode commands for core register access
- GO command to resume execution
- BACKGROUND command to halt core or wake CPU from low-power modes
- Oscillator runs in stop mode, if BDM enabled

Based on these features, BDM is useful for the following reasons:

- In-circuit emulation is not needed, so physical and electrical characteristics of the system are not affected.
- BDM is always available for debugging the system and provides a communication link for upgrading firmware in existing systems.
- Provides high-speed memory downloading, especially useful for flash programming
- Provides absolute control of the processor, and thus the system. This feature allows quick hardware debugging with the same tool set used for firmware development.

18.4.1.1 CPU Halt

Although certain BDM operations can occur in parallel with CPU operations, unrestricted BDM operation requires the CPU to be halted. The sources that can cause the CPU to halt are listed below in order of priority. Recall that the default configuration of the Version 1 ColdFire core (CF1Core) defines the occurrence of certain exception types to automatically generate a system reset. Some of these fault types include illegal instructions, privilege errors, address errors, and bus error terminations, with the response under control of the processor's CPUCR[ARD, IRD] bits.

Table 18-23. CPU Halt Sources

Halt Source	Halt Timing	Description	
Fault-on-fault	Immediate	Refers to the occurrence of any fault while exception processing. For example, a bus error is signaled during exception stack frame writes or while fetching the first instruction in the exception service routine.	
		CPUCR[ARD] = 1	Immediately enters halt.
		CPUCR[ARD] = 0	Reset event is initiated.

MCF51QE128 MCU Series Reference Manual, Rev. 3

Table 18-23. CPU Halt Sources (continued)

Halt Source	Halt Timing	Description		
Hardware breakpoint trigger	Pending	Halt is made pending in the processor. The processor samples for pending halt and interrupt conditions once per instruction. When a pending condition is asserted, the processor halts execution at the next sample point.		
HALT instruction	Immediate	BDM disabled	CPUCR[IRD] = 0	A reset is initiated since attempted execution of an illegal instruction
			CPUCR[IRD] = 1	An illegal instruction exception is generated.
		BDM enabled, supervisor mode	Processor immediately halts execution at the next instruction sample point. The processor can be restarted by a BDM GO command. Execution continues at the instruction after HALT.	
		BDM enabled, user mode	CSR[UHE] = 0 CPUCR[IRD] = 0	A reset event is initiated, since a privileged instruction was attempted in user mode.
			CSR[UHE] = 0 CPUCR[IRD] = 1	A privilege violation exception is generated.
			CSR[UHE] = 1	Processor immediately halts execution at the next instruction sample point. The processor can be restarted by a BDM GO command. Execution continues at the instruction after HALT.
BACKGROUND command	Pending	BDM disabled or flash secure	Illegal command response and BACKGROUND command is ignored.	
		BDM enabled and flash unsecure	Processor is running	Halt is made pending in the processor. The processor samples for pending halt and interrupt conditions once per instruction. When a pending condition is asserted, the processor halts execution at the next sample point.
			Processor is stopped	Processing of the BACKGROUND command is treated in a special manner. The processor exits the stopped mode and enters the halted state, at which point all BDM commands may be exercised. When restarted, the processor continues by executing the next sequential instruction (the instruction following STOP).

Table 18-23. CPU Halt Sources (continued)

Halt Source	Halt Timing		Description
BKGD held low for ≥2 bus clocks after reset negated for POR or BDM reset	Immediate	Flash unsecure	Enters debug mode with XCSR[ENBDM, CLKSW] set. The full set of BDM commands is available and debug can proceed. If the core is reset into a debug halt condition, the processor's response to the GO command depends on the BDM command(s) performed while it was halted. Specifically, if the PC register was loaded, the GO command causes the processor to exit halted state and pass control to the instruction address in the PC, bypassing normal reset exception processing. If the PC was not loaded, the GO command causes the processor to exit halted state and continue reset exception processing.
		Flash secure	Enters debug mode with XCSR[ENBDM, CLKSW] set. The allowable commands are limited to the always-available group. A GO command to start the processor is not allowed. The only recovery actions in this mode are: • Issue a BDM reset setting CSR2[BDFR] with CSR2[BDHBR] cleared and the BKGD pin held high • Erase the flash to unsecure the memory and then proceed with debug • Power cycle the device with the BKGD pin held high to reset into the normal operating mode

The processor's run/stop/halt status is always accessible in XCSR[CPUHALT,CPUSTOP]. Additionally, CSR[27–24] indicate the halt source, showing the highest priority source for multiple halt conditions. This field is cleared by a read of the CSR. The debug GO command also clears CSR[26–24].

18.4.1.2 Background Debug Serial Interface Controller (BDC)

BDC serial communications use a custom serial protocol first introduced on the M68HC12 Family of microcontrollers and later used in the M68HCS08 family. This protocol assumes that the host knows the communication clock rate determined by the target BDC clock rate. The BDC clock rate may be the system bus clock frequency or an alternate frequency source depending on the state of XCSR[CLKSW]. All communication is initiated and controlled by the host which drives a high-to-low edge to signal the beginning of each bit time. Commands and data are sent most significant bit (msb) first. For a detailed description of the communications protocol, refer to Section 18.4.1.3, "BDM Communication Details".

If a host is attempting to communicate with a target MCU that has an unknown BDC clock rate, a SYNC command may be sent to the target MCU to request a timed synchronization response signal from which the host can determine the correct communication speed. After establishing communications, the host can read XCSR and write the clock switch (CLKSW) control bit to change the source of the BDC clock for further serial communications if necessary.

BKGD is a pseudo-open-drain pin and there is an on-chip pullup so no external pullup resistor is required. Unlike typical open-drain pins, the external RC time constant on this pin, which is influenced by external capacitance, plays almost no role in signal rise time. The custom protocol provides for brief, actively driven speed-up pulses to force rapid rise times on this pin without risking harmful drive level conflicts. Refer to Section 18.4.1.3, "BDM Communication Details," for more details.

When no debugger pod is connected to the standard 6-pin BDM interface connector (Section 18.4.4, "Freescale-Recommended BDM Pinout"), the internal pullup on BKGD chooses normal operating mode.

When a development system is connected, it can pull BKGD and $\overline{\text{RESET}}$ low, release $\overline{\text{RESET}}$ to select active background (halt) mode rather than normal operating mode, and then release BKGD. It is not necessary to reset the target MCU to communicate with it through the background debug interface. There is also a mechanism to generate a reset event in response to setting CSR2[BDFR].

18.4.1.3   BDM Communication Details

The BDC serial interface requires the external host controller to generate a falling edge on the BKGD pin to indicate the start of each bit time. The external controller provides this falling edge whether data is transmitted or received.

BKGD is a pseudo-open-drain pin that can be driven by an external controller or by the MCU. Data is transferred msb first at 16 BDC clock cycles per bit (nominal speed). The interface times-out if 512 BDC clock cycles occur between falling edges from the host. If a time-out occurs, the status of any command in progress must be determined before new commands can be sent from the host. To check the status of the command, follow the steps detailed in the bit description of XCSR[CSTAT] in Table 18-6.

The custom serial protocol requires the debug pod to know the target BDC communication clock speed. The clock switch (CLKSW) control bit in the XCSR[31–24] register allows you to select the BDC clock source. The BDC clock source can either be the bus clock or the alternate BDC clock source. When the MCU is reset in normal user mode, CLKSW is cleared and that selects the alternate clock source. This clock source is a fixed frequency independent of the bus frequency so it does change if the user modifies clock generator settings. This is the preferred clock source for general debugging.

When the MCU is reset in active background (halt) mode, CLKSW is set which selects the bus clock as the source of the BDC clock. This CLKSW setting is most commonly used during flash memory programming because the bus clock can usually be configured to operate at the highest allowed bus frequency to ensure the fastest possible flash programming times. Since the host system is in control of changes to clock generator settings, it knows when a different BDC communication speed should be used. The host programmer also knows that no unexpected change in bus frequency could occur to disrupt BDC communications.

Normally, setting CLKSW should not be used for general debugging because there is no way to ensure the application program does not change the clock generator settings. This is especially true in the case of application programs that are not yet fully debugged.

After any reset (or at any other time), the host system can issue a SYNC command to determine the speed of the BDC clock. CLKSW may be written using the serial WRITE_XCSR_BYTE command through the BDC interface. CLKSW is located in the special XCSR byte register in the BDC module and it is not accessible in the normal memory map of the ColdFire core. This means that no program running on the processor can modify this register (intentionally or unintentionally).

The BKGD pin can receive a high- or low-level or transmit a high- or low-level. The following diagrams show timing for each of these cases. Interface timing is synchronous to clocks in the target BDC, but asynchronous to the external host. The internal BDC clock signal is shown for reference in counting cycles.

Figure 18-15 shows an external host transmitting a logic 1 or 0 to the BKGD pin of a target MCU. The host is asynchronous to the target so there is a 0–1 cycle delay from the host-generated falling edge to

where the target perceives the beginning of the bit time. Ten target BDC clock cycles later, the target senses the bit level on the BKGD pin. Typically, the host actively drives the pseudo-open-drain BKGD pin during host-to-target transmissions to speed up rising edges. Because the target does not drive the BKGD pin during the host-to-target transmission period, there is no need to treat the line as an open-drain signal during this period.
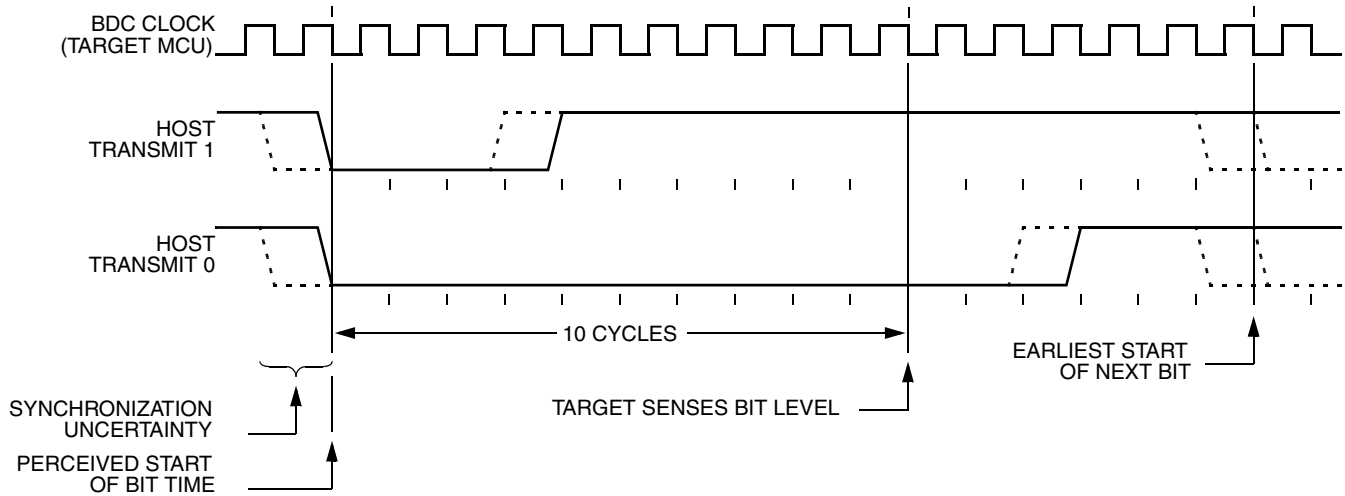
Figure 18-15. BDC Host-to-Target Serial Bit Timing

Figure 18-16 shows the host receiving a logic 1 from the target MCU. Because the host is asynchronous to the target MCU, there is a 0–1 cycle delay from the host-generated falling edge on BKGD to the perceived start of the bit time in the target MCU. The host holds the BKGD pin low long enough for the target to recognize it (at least two target BDC cycles). The host must release the low drive before the target MCU drives a brief active-high speedup pulse seven cycles after the perceived start of the bit time. The host should sample the bit level about 10 cycles after it started the bit time.
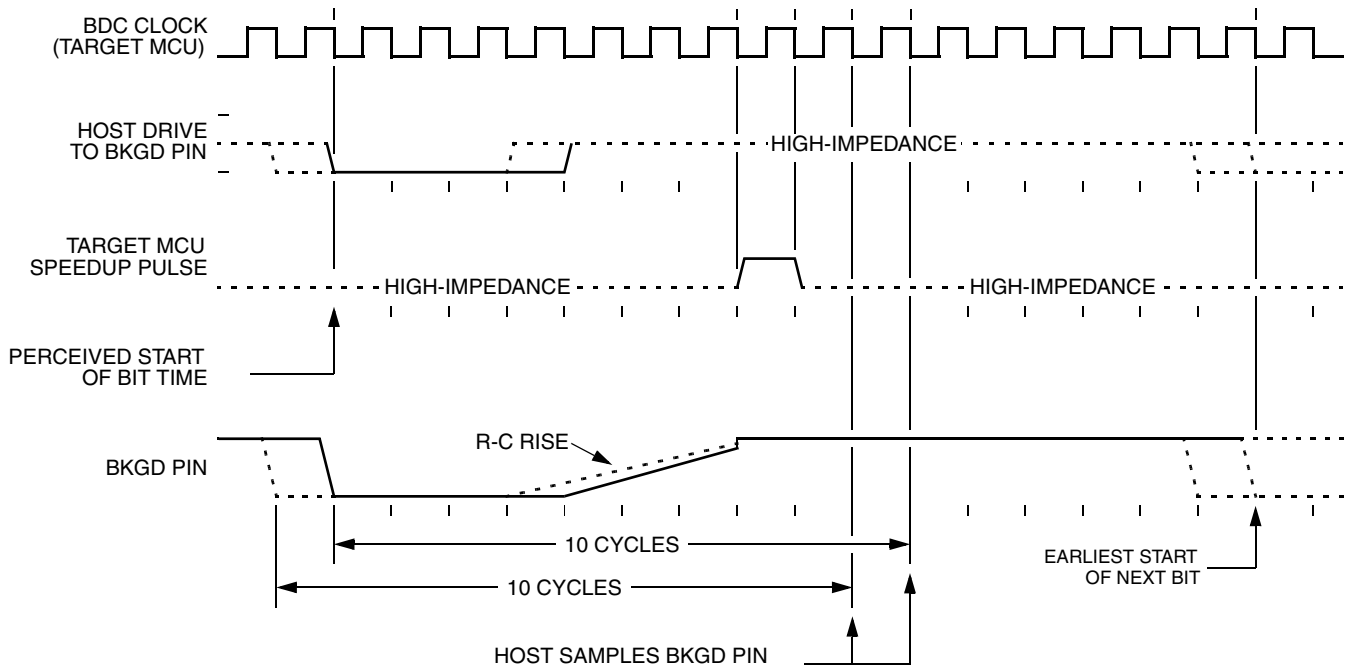
Figure 18-16. BDC Target-to-Host Serial Bit Timing (Logic 1)

Figure 18-17 shows the host receiving a logic 0 from the target MCU. Because the host is asynchronous to the target MCU, there is a 0–1 cycle delay from the host-generated falling edge on BKGD to the start of the bit time as perceived by the target MCU. The host initiates the bit time, but the target MCU finishes it. Because the target wants the host to receive a logic 0, it drives the BKGD pin low for 13 BDC clock cycles, then briefly drives it high to speed up the rising edge. The host samples the bit level about 10 cycles after starting the bit time.
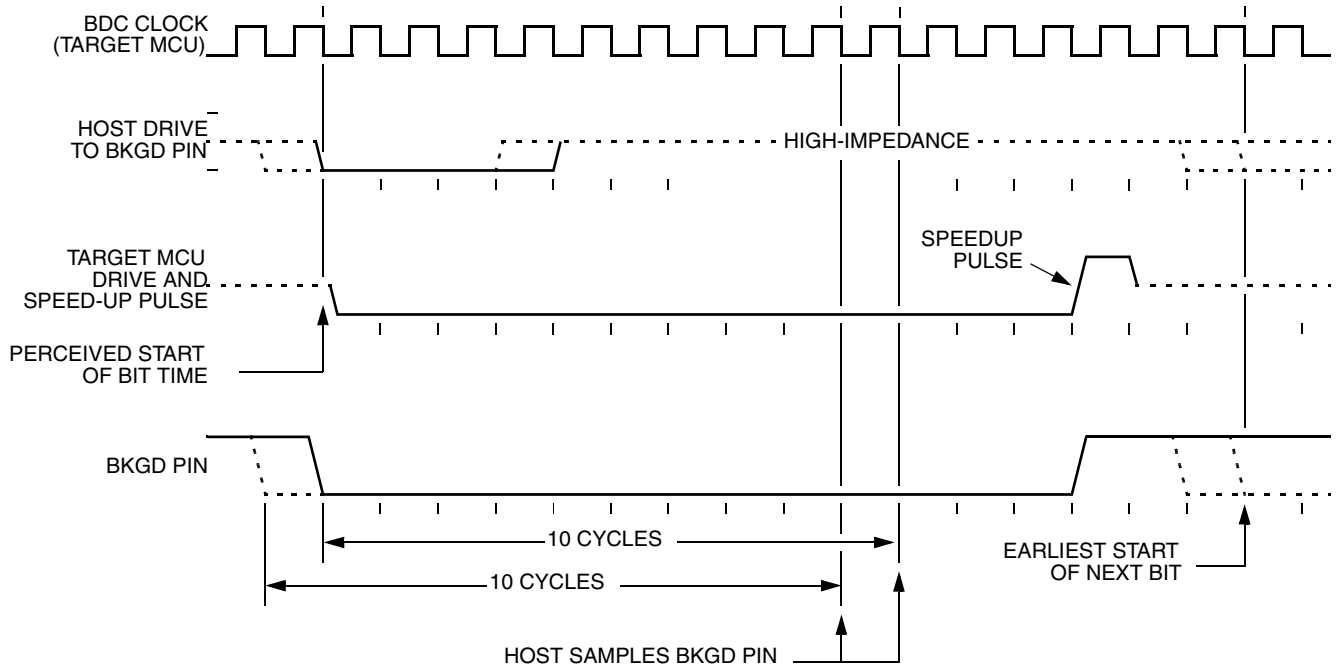
Figure 18-17. BDM Target-to-Host Serial Bit Timing (Logic 0)

18.4.1.4   BDM Command Set Descriptions

This section presents detailed descriptions of the BDM commands.

The V1 BDM command set is based on transmission of one or more 8-bit data packets per operation. Each operation begins with a host-to-target transmission of an 8-bit command code packet. The command code definition broadly maps the operations into four formats as shown in Figure 18-18.

Miscellaneous Commands

	7	6	5	4	3	2	1	0
W	0	0	R/$\overline{W}$	0	MSCMD
R/W	Optional Command Extension Byte (Data)

Memory Commands

	7	6	5	4	3	2	1	0
W	0	0	R/$\overline{W}$	1	\multicolumn{2}{c}{SZ}	\multicolumn{2}{c}{MCMD}
W if addr, R/W if data	Command Extension Bytes (Address, Data)

Core Register Commands

	7	6	5	4	3	2	1	0
W	\multicolumn{2}{c}{CRG}	R/$\overline{W}$	\multicolumn{5}{c}{CRN}
R/W	Command Extension Bytes (Data)

PST Trace Buffer Read Commands

	7	6	5	4	3	2	1	0
W	0	1	0	1	CRN
R	Trace Buffer Data[31–24]
R	Trace Buffer Data[23–16]
R	Trace Buffer Data[15–08]
R	Trace Buffer Data[07–00]

Figure 18-18. BDM Command Code Encodings

Table 18-24. BDM Command Code Field Descriptions

Field	Description
5 R/$\overline{W}$	Read/$\overline{Write}$. 0 Command is performing a write operation. 1 Command is performing a read operation.
3–0 MSCMD	Miscellaneous command. Defines the miscellaneous command to be performed. 0000 No operation 0001 Display the CPU's program counter (PC) plus optional capture in the PST trace buffer 0010 Enable the BDM acknowledge communication mode 0011 Disable the BDM acknowledge communication mode 0100 Force a CPU halt (background) 1000 Resume CPU execution (go) 1101 Read/write of the debug XCSR most significant byte 1110 Read/write of the debug CSR2 most significant byte 1111 Read/write of the debug CSR3 most significant byte
3–2 SZ	Memory operand size. Defines the size of the memory reference. 00 8-bit byte 01 16-bit word 10 32-bit long
1–0 MCMD	Memory command. Defines the type of the memory reference to be performed. 00 Simple write if R/$\overline{W}$ = 0; simple read if R/$\overline{W}$ = 1 01 Write + status if R/$\overline{W}$ = 0; read + status if R/$\overline{W}$ = 1 10 Fill if R/$\overline{W}$ = 0; dump if R/$\overline{W}$ = 1 11 Fill + status if R/$\overline{W}$ = 0; dump + status if R/$\overline{W}$ = 1
7–6 CRG	Core register group. Defines the core register group to be referenced. 01 CPU's general-purpose registers (An, Dn) or PST trace buffer 10 DBG's control registers 11 CPU's control registers (PC, SR, VBR, CPUCR,...)
4–0 CRN	Core register number. Defines the specific core register (its number) to be referenced. All other CRN values are reserved. <table><tr><th>CRG</th><th>CRN</th><th>Register</th></tr><tr><td rowspan="3">01</td><td>0x00–0x07</td><td>D0–7</td></tr><tr><td>0x08–0x0F</td><td>A0–7</td></tr><tr><td>0x10–0x1B</td><td>PST Buffer 0–11</td></tr><tr><td>10</td><td colspan="2">DRc[4:0] as described in Table 18-4</td></tr><tr><td rowspan="5">11</td><td>0x00</td><td>OTHER_A7</td></tr><tr><td>0x01</td><td>VBR</td></tr><tr><td>0x02</td><td>CPUCR</td></tr><tr><td>0x0E</td><td>SR</td></tr><tr><td>0x0F</td><td>PC</td></tr></table>

18.4.1.5   BDM Command Set Summary

Table 18-25 summarizes the BDM command set. Subsequent paragraphs contain detailed descriptions of each command. The nomenclature below is used in Table 18-25 to describe the structure of the BDM commands.

Commands begin with an 8-bit hexadecimal command code in the host-to-target direction (most significant bit first)

```
/        =        separates parts of the command
d        =        delay 16 target BDC clock cycles
ad24     =        24-bit memory address in the host-to-target direction
rd8      =        8 bits of read data in the target-to-host direction
rd16     =        16 bits of read data in the target-to-host direction
rd32     =        32 bits of read data in the target-to-host direction
rd.sz    =        read data, size defined by sz, in the target-to-host direction
wd8      =        8 bits of write data in the host-to-target direction
wd16     =        16 bits of write data in the host-to-target direction
wd32     =        32 bits of write data in the host-to-target direction
wd.sz    =        write data, size defined by sz, in the host-to-target direction
ss       =        the contents of XCSR[31:24] in the target-to-host direction (STATUS)
sz       =        memory operand size (0b00 = byte, 0b01 = word, 0b10 = long)
crn      =        core register number
WS       =        command suffix signaling the operation is "with status"
```

Table 18-25. BDM Command Summary

Command Mnemonic	Command Classification	ACK if Enb?[1]	Command Structure	Description
SYNC	Always Available	N/A	N/A[2]	Request a timed reference pulse to determine the target BDC communication speed
ACK_DISABLE	Always Available	No	0x03/d	Disable the communication handshake. This command does not issue an ACK pulse.
ACK_ENABLE	Always Available	Yes	0x02/d	Enable the communication handshake. Issues an ACK pulse after the command is executed.
BACKGROUND	Non-Intrusive	Yes	0x04/d	Halt the CPU if ENBDM is set. Otherwise, ignore as illegal command.
DUMP_MEM.sz	Non-Intrusive	Yes	(0x32+4 x sz)/d/rd.sz	Dump (read) memory based on operand size (sz). Used with READ_MEM to dump large blocks of memory. An initial READ_MEM is executed to set up the starting address of the block and to retrieve the first result. Subsequent DUMP_MEM commands retrieve sequential operands.

Table 18-25. BDM Command Summary (continued)

Command Mnemonic	Command Classification	ACK if Enb?[1]	Command Structure	Description
DUMP_MEM.sz_WS	Non-Intrusive	No	(0x33+4 x sz)/d/ss/rd.sz	Dump (read) memory based on operand size (sz) and report status. Used with READ_MEM{_WS} to dump large blocks of memory. An initial READ_MEM{_WS} is executed to set up the starting address of the block and to retrieve the first result. Subsequent DUMP_MEM{_WS} commands retrieve sequential operands.
FILL_MEM.sz	Non-Intrusive	Yes	(0x12+4 x sz)/wd.sz/d	Fill (write) memory based on operand size (sz). Used with WRITE_MEM to fill large blocks of memory. An initial WRITE_MEM is executed to set up the starting address of the block and to write the first operand. Subsequent FILL_MEM commands write sequential operands.
FILL_MEM.sz_WS	Non-Intrusive	No	(0x13+4 x sz)/wd.sz/d/ss	Fill (write) memory based on operand size (sz) and report status. Used with WRITE_MEM{_WS} to fill large blocks of memory. An initial WRITE_MEM{_WS} is executed to set up the starting address of the block and to write the first operand. Subsequent FILL_MEM{_WS} commands write sequential operands.
GO	Non-Intrusive	Yes	0x08/d	Resume the CPU's execution[3]
NOP	Non-Intrusive	Yes	0x00/d	No operation
READ_CREG	Active Background	Yes	(0xE0+CRN)/d/rd32	Read one of the CPU's control registers
READ_DREG	Non-Intrusive	Yes	(0xA0+CRN)/d/rd32	Read one of the debug module's control registers
READ_MEM.sz	Non-Intrusive	Yes	(0x30+4 x sz)/ad24/d/rd.sz	Read the appropriately-sized (sz) memory value from the location specified by the 24-bit address
READ_MEM.sz_WS	Non-Intrusive	No	(0x31+4 x sz)/ad24/d/ss/rd.sz	Read the appropriately-sized (sz) memory value from the location specified by the 24-bit address and report status
READ_PSTB	Non-Intrusive	Yes	(0x50+CRN)/d/rd32	Read the requested longword location from the PST trace buffer
READ_Rn	Active Background	Yes	(0x60+CRN)/d/rd32	Read the requested general-purpose register (An, Dn) from the CPU
READ_XCSR_BYTE	Always Available	No	0x2D/rd8	Read the most significant byte of the debug module's XCSR
READ_CSR2_BYTE	Always Available	No	0x2E/rd8	Read the most significant byte of the debug module's CSR2
READ_CSR3_BYTE	Always Available	No	0x2F/rd8	Read the most significant byte of the debug module's CSR3

MCF51QE128 MCU Series Reference Manual, Rev. 3

Table 18-25. BDM Command Summary (continued)

Command Mnemonic	Command Classification	ACK if Enb?[1]	Command Structure	Description
SYNC_PC	Non-Intrusive	Yes	0x01/d	Display the CPU's current PC and capture it in the PST trace buffer
WRITE_CREG	Active Background	Yes	(0xC0+CRN)/wd32/d	Write one of the CPU's control registers
WRITE_DREG	Non-Intrusive	Yes	(0x80+CRN)/wd32/d	Write one of the debug module's control registers
WRITE_MEM.sz	Non-Intrusive	Yes	(0x10+4 x sz)/ad24/wd.sz/d	Write the appropriately-sized (sz) memory value to the location specified by the 24-bit address
WRITE_MEM.sz_WS	Non-Intrusive	No	(0x11+4 x sz)/ad24/wd.sz/d/ss	Write the appropriately-sized (sz) memory value to the location specified by the 24-bit address and report status
WRITE_Rn	Active Background	Yes	(0x40+CRN)/wd32/d	Write the requested general-purpose register (An, Dn) of the CPU
WRITE_XCSR_BYTE	Always Available	No	0x0D/wd8	Write the most significant byte of the debug module's XCSR
WRITE_CSR2_BYTE	Always Available	No	0x0E/wd8	Write the most significant byte of the debug module's CSR2
WRITE_CSR3_BYTE	Always Available	No	0x0F/wd8	Write the most significant byte of the debug module's CSR3

[1] This column identifies if the command generates an ACK pulse if operating with acknowledge mode enabled. See Section 18.4.1.7, "Hardware Handshake Abort Procedure," for addition information.

[2] The SYNC command is a special operation which does not have a command code.

[3] If a GO command is received while the processor is not halted, it performs no operation.

18.4.1.5.1   SYNC

The SYNC command is unlike other BDC commands because the host does not necessarily know the correct speed to use for serial communications until after it has analyzed the response to the SYNC command.

To issue a SYNC command, the host:

1. Drives the BKGD pin low for at least 128 cycles of the slowest possible BDC clock (bus clock or device-specific alternate clock source).
2. Drives BKGD high for a brief speed-up pulse to get a fast rise time. (This speedup pulse is typically one cycle of the host clock which is as fast as the maximum target BDC clock.)
3. Removes all drive to the BKGD pin so it reverts to high impedance.
4. Listens to the BKGD pin for the sync response pulse.

Upon detecting the sync request from the host (which is a much longer low time than would ever occur during normal BDC communications), the target:

1. Waits for BKGD to return to a logic high.

2. Delays 16 cycles to allow the host to stop driving the high speed-up pulse.

3. Drives BKGD low for 128 BDC clock cycles.

4. Drives a 1-cycle high speed-up pulse to force a fast rise time on BKGD.

5. Removes all drive to the BKGD pin so it reverts to high impedance.

The host measures the low time of this 128-cycle sync response pulse and determines the correct speed for subsequent BDC communications. Typically, the host can determine the correct communication speed within a few percent of the actual target speed and the serial protocol can easily tolerate this speed error.

18.4.1.5.2   ACK_DISABLE

Disable host/target handshake protocol   **Always Available**

```
┌───────────┬──┐
│   0x03    │  │
└───────────┴──┘
  host →       D
  target       L
               Y
```
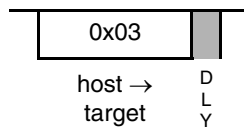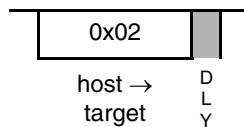
Disables the serial communication handshake protocol. The subsequent commands, issued after the ACK_DISABLE command, do not execute the hardware handshake protocol. This command is not followed by an ACK pulse.

18.4.1.5.3   ACK_ENABLE

Enable host/target handshake protocol   **Always Available**

```
┌───────────┬──┐
│   0x02    │  │
└───────────┴──┘
  host →       D
  target       L
               Y
```

Enables the hardware handshake protocol in the serial communication. The hardware handshake is implemented by an acknowledge (ACK) pulse issued by the target MCU in response to a host command. The ACK_ENABLE command is interpreted and executed in the BDC logic without the need to interface with the CPU. However, an acknowledge (ACK) pulse is issued by the target device after this command is executed. This feature can be used by the host in order to evaluate if the target supports the hardware handshake protocol. If the target supports the hardware handshake protocol, subsequent commands are enabled to execute the hardware handshake protocol, otherwise this command is ignored by the target.

For additional information about the hardware handshake protocol, refer to Section 18.4.1.6, "Serial Interface Hardware Handshake Protocol," and Section 18.4.1.7, "Hardware Handshake Abort Procedure."

18.4.1.5.4   BACKGROUND

Enter active background mode (if enabled)   **Non-intrusive**

```
      ┌──────────┐▐
      │   0x04   │▐
      └──────────┘▐

        host →    D
        target    L
                  Y
```
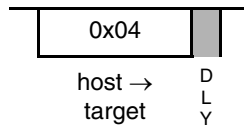
Provided XCSR[ENBDM] is set (BDM enabled), the BACKGROUND command causes the target MCU to enter active background (halt) mode as soon as the current CPU instruction finishes. If ENBDM is cleared (its default value), the BACKGROUND command is ignored.

A delay of 16 BDC clock cycles is required after the BACKGROUND command to allow the target MCU to finish its current CPU instruction and enter active background mode before a new BDC command can be accepted.
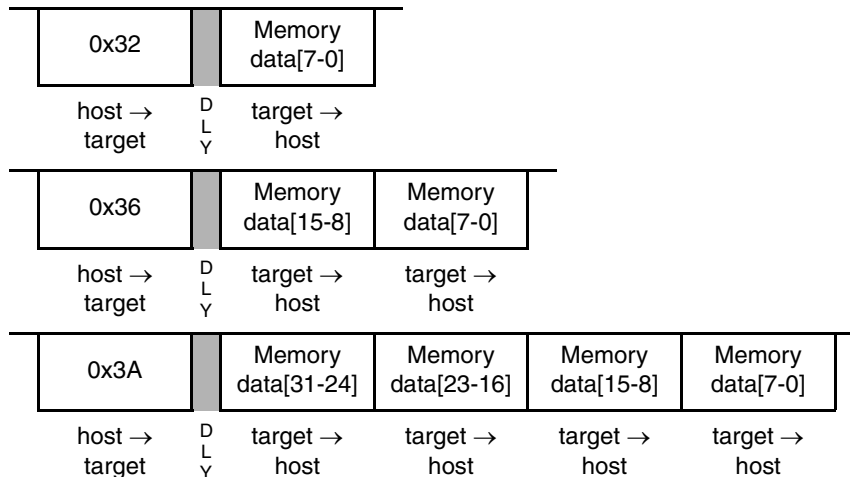
After the target MCU is reset into a normal operating mode, the host debugger would send a WRITE_XCSR_BYTE command to set ENBDM before attempting to send the BACKGROUND command the first time. Normally, the development host would set ENBDM once at the beginning of a debug session or after a target system reset, and then leave the ENBDM bit set during debugging operations. During debugging, the host would use GO commands to move from active background mode to normal user program execution and would use BACKGROUND commands or breakpoints to return to active background mode.

18.4.1.5.5   DUMP_MEM.sz, DUMP_MEM.sz_WS

DUMP_MEM.sz

Read memory specified by debug address register, then   **Non-intrusive**
increment address

```
   ┌──────────┐▐ ┌──────────┐
   │   0x32   │▐ │ Memory   │
   │          │▐ │ data[7-0]│
   └──────────┘▐ └──────────┘
     host →    D   target →
     target    L    host
               Y

   ┌──────────┐▐ ┌──────────┐┌──────────┐
   │   0x36   │▐ │ Memory   ││ Memory   │
   │          │▐ │data[15-8]││ data[7-0]│
   └──────────┘▐ └──────────┘└──────────┘
     host →    D   target →   target →
     target    L    host        host
               Y

   ┌──────────┐▐ ┌───────────┐┌───────────┐┌──────────┐┌──────────┐
   │   0x3A   │▐ │  Memory   ││  Memory   ││ Memory   ││ Memory   │
   │          │▐ │data[31-24]││data[23-16]││data[15-8]││ data[7-0]│
   └──────────┘▐ └───────────┘└───────────┘└──────────┘└──────────┘
     host →    D   target →    target →     target →    target →
     target    L    host         host         host        host
               Y
```

Get the latest version from freescale.com

DUMP_MEM.sz_WS

Read memory specified by debug address register with status, then increment address   **Non-intrusive**

0x33		XCSR_SB	Memory data[7-0]
host → target	D L Y	target → host	target → host

0x37		XCSR_SB	Memory data[15-8]	Memory data[7-0]
host → target	D L Y	target → host	target → host	target → host

0x3B		XCSR_SB	Memory data[31-24]	Memory data23-16]	Memory data[15-8]	Memory data[7-0]
host → target	D L Y	target → host	target → host	target → host	target → host	target → host
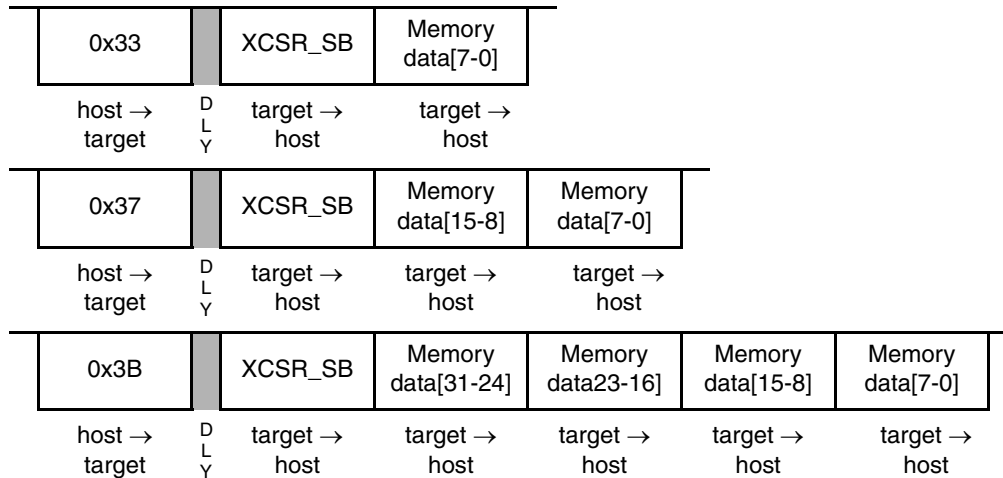
DUMP_MEM{_WS} is used with the READ_MEM{_WS} command to access large blocks of memory. An initial READ_MEM{_WS} is executed to set-up the starting address of the block and to retrieve the first result. If an initial READ_MEM{_WS} is not executed before the first DUMP_MEM{_WS}, an illegal command response is returned. The DUMP_MEM{_WS} command retrieves subsequent operands. The initial address is incremented by the operand size (1, 2, or 4) and saved in a temporary register. Subsequent DUMP_MEM{_WS} commands use this address, perform the memory read, increment it by the current operand size, and store the updated address in the temporary register. If the with-status option is specified, the core status byte contained in XCSR[31–24] (XCSR_SB) is returned before the read data. The XCSR status byte reflects the state after the memory read was performed.
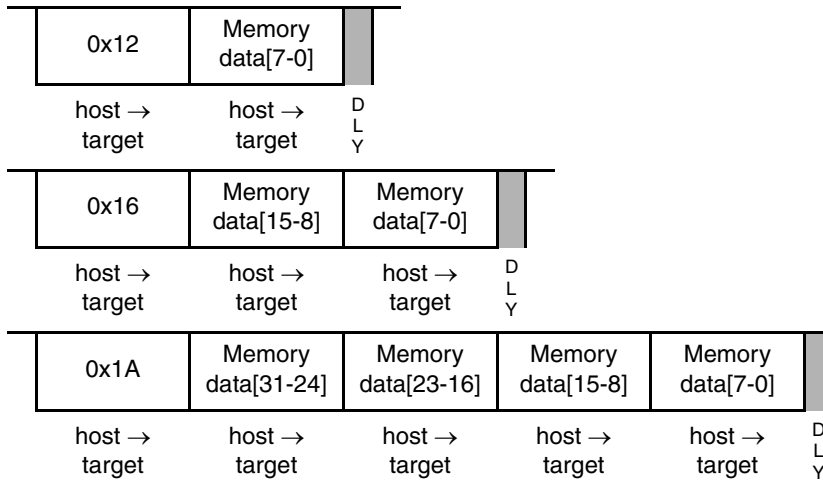
NOTE

DUMP_MEM_{WS} does not check for a valid address; it is a valid command only when preceded by NOP, READ_MEM_{WS}, or another DUMP_MEM{_WS} command. Otherwise, an illegal command response is returned. NOP can be used for inter-command padding without corrupting the address pointer.

The size field (sz) is examined each time a DUMP_MEM{_WS} command is processed, allowing the operand size to be dynamically altered. The examples show the DUMP_MEM.B{_WS}, DUMP_MEM.W{_WS} and DUMP_MEM.L{_WS} commands.

18.4.1.5.6   FILL_MEM.sz, FILL_MEM.sz_WS

FILL_MEM.sz

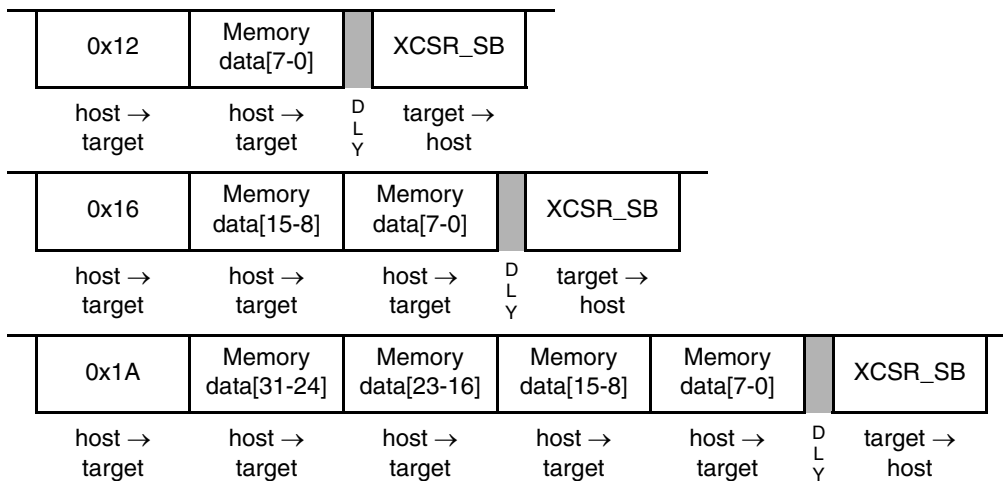**Write memory specified by debug address register, then increment address**   **Non-intrusive**

0x12	Memory data[7-0]	D L Y
host → target	host → target	

0x16	Memory data[15-8]	Memory data[7-0]	D L Y
host → target	host → target	host → target	

0x1A	Memory data[31-24]	Memory data[23-16]	Memory data[15-8]	Memory data[7-0]	D L Y
host → target	host → target	host → target	host → target	host → target	

FILL_MEM.sz_WS

Write memory specified by debug address register with status, then increment address   **Non-intrusive**

0x12	Memory data[7-0]	D L Y	XCSR_SB
host → target	host → target		target → host

0x16	Memory data[15-8]	Memory data[7-0]	D L Y	XCSR_SB
host → target	host → target	host → target		target → host

| 0x1A	Memory data[31-24]	Memory data[23-16]	Memory data[15-8]	Memory data[7-0]	D L Y	XCSR_SB
host → target	host → target	host → target	host → target	host → target		target → host

FILL_MEM{_WS} is used with the WRITE_MEM{_WS} command to access large blocks of memory. An initial WRITE_MEM{_WS} is executed to set up the starting address of the block and write the first datum. If an initial WRITE_MEM{_WS} is not executed before the first FILL_MEM{_WS}, an illegal command response is returned. The FILL_MEM{_WS} command stores subsequent operands. The initial address is incremented by the operand size (1, 2, or 4) and saved in a temporary register. Subsequent WRITE_MEM{_WS} commands use this address, perform the memory write, increment it by the current operand size, and store the updated address in the temporary register. If the with-status option is specified, the core status byte contained in XCSR[31–24] (XCSR_SB) is returned after the write data. The XCSR status byte reflects the state after the memory write was performed.

NOTE

FILL_MEM_{WS} does not check for a valid address; it is a valid command only when preceded by NOP, WRITE_MEM_{WS}, or another FILL_MEM{_WS} command. Otherwise, an illegal command response is returned. NOP can be used for intercommand padding without corrupting the address pointer.
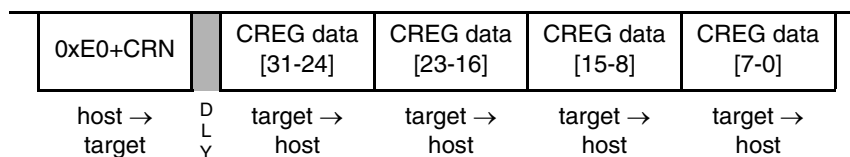
The size field (sz) is examined each time a FILL_MEM{_WS} command is processed, allowing the operand size to be dynamically altered. The examples show the FILL_MEM.B{_WS}, FILL_MEM.W{_WS} and FILL_MEM.L{_WS} commands.

18.4.1.5.7   GO

Go   **Non-intrusive**

0x08	
host → target	D L Y
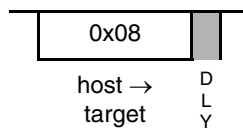
This command is used to exit active background (halt) mode and begin (or resume) execution of the application's instructions. The CPU's pipeline is flushed and refilled before normal instruction execution resumes. Prefetching begins at the current address in the PC and at the current privilege level. If any register (such as the PC or SR) is altered by a BDM command while the processor is halted, the updated value is used when prefetching resumes. If a GO command is issued and the CPU is not halted, the command is ignored.

18.4.1.5.8   NOP

No operation   **Non-intrusive**

0x00	
host → target	D L Y
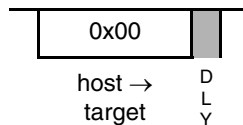
NOP performs no operation and may be used as a null command where required.

18.4.1.5.9   READ_CREG

Read CPU control register   **Active Background**

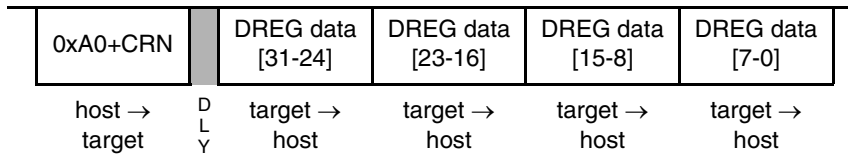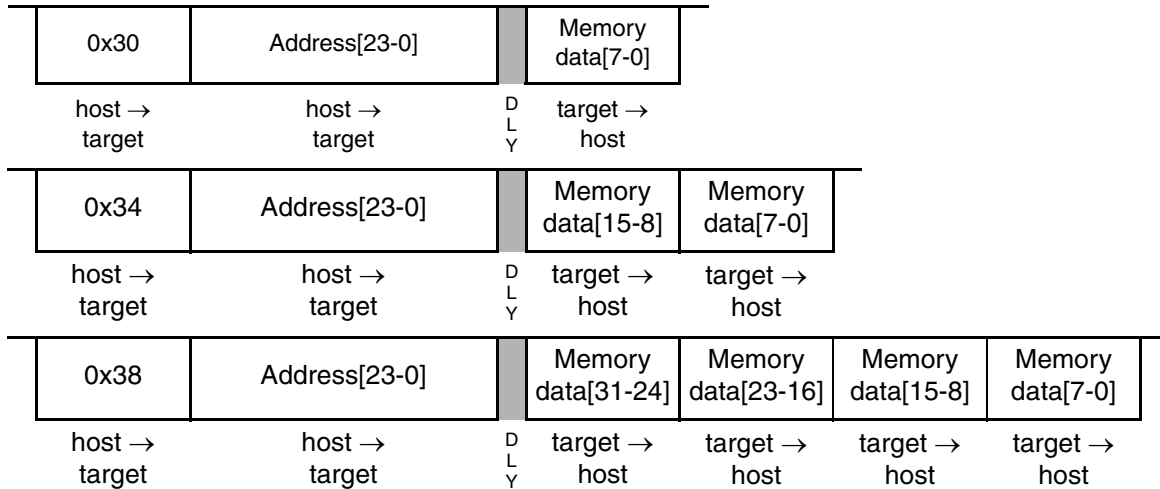| 0xE0+CRN		CREG data [31-24]	CREG data [23-16]	CREG data [15-8]	CREG data [7-0]
host → target	D L Y	target → host	target → host	target → host	target → host

If the processor is halted, this command reads the selected control register and returns the 32-bit result. This register grouping includes the PC, SR, CPUCR, VBR, and OTHER_A7. Accesses to processor control registers are always 32-bits wide, regardless of implemented register width. The register is addressed through the core register number (CRN). See Table 18-24 for the CRN details when CRG is 11.

If the processor is not halted, this command is rejected as an illegal operation and no operation is performed.

18.4.1.5.10   READ_DREG

Read debug control register   **Non-intrusive**

0xA0+CRN		DREG data [31-24]	DREG data [23-16]	DREG data [15-8]	DREG data [7-0]
host → target	D L Y	target → host	target → host	target → host	target → host

This command reads the selected debug control register and returns the 32-bit result. This register grouping includes the CSR, XCSR, CSR2, and CSR3. Accesses to debug control registers are always 32-bits wide, regardless of implemented register width. The register is addressed through the core register number (CRN). See Table 18-4 for CRN details.

18.4.1.5.11   READ_MEM.sz, READ_MEM.sz_WS

READ_MEM.sz

Read memory at the specified address   **Non-intrusive**

0x30	Address[23-0]		Memory data[7-0]
host → target	host → target	D L Y	target → host

0x34	Address[23-0]		Memory data[15-8]	Memory data[7-0]
host → target	host → target	D L Y	target → host	target → host

0x38	Address[23-0]		Memory data[31-24]	Memory data[23-16]	Memory data[15-8]	Memory data[7-0]
host → target	host → target	D L Y	target → host	target → host	target → host	target → host

READ_MEM.sz_WS

Read memory at the specified address with status   **Non-intrusive**

0x31	Address[23-0]	D L Y	XCSR_SB	Memory data[7-0]
host → target	host → target		target → host	target → host

0x35	Address[23-0]	D L Y	XCSR_SB	Memory data [15-8]	Memory data [7-0]
host → target	host → target		target → host	target → host	target → host

0x39	Address[23-0]	D L Y	XCSR_SB	Memory data[31-24]	Memory data[23-16]	Memory data [15-8]	Memory data [7-0]
host → target	host → target		target → host	target → host	target → host	target → host	target → host

Read data at the specified memory address. The reference address is transmitted as three 8-bit packets (msb to lsb) immediately after the command packet. The access attributes are defined by BAAR[TT,TM]. The hardware forces low-order address bits to zeros for word and longword accesses to ensure these accesses are on 0-modulo-size alignments. If the with-status option is specified, the core status byte contained in XCSR[31–24] (XCSR_SB) is returned before the read data. The XCSR status byte reflects the state after the memory read was performed.

The examples show the READ_MEM.B{_WS}, READ_MEM.W{_WS} and READ_MEM.L{_WS} commands.

18.4.1.5.12   READ_PSTB

Read PST trace buffer at the specified address   **Non-intrusive**

| 0x50+CRN	D L Y	PSTB data [31-24]	PSTB data [23-16]	PSTB data [15-8]	PSTB data [7-0]
host → target		target → host	target → host	target → host	target → host

Read 32 bits of captured PST/DDATA values from the trace buffer at the specified address. The PST trace buffer contains 64 six-bit entries, packed consecutively into 12 longword locations. See Table 18-24 for the CRN details when CRG is 01.

The write pointer for the trace buffer is available as CSR2[PSTBWA]. Using this pointer, it is possible to determine the oldest-to-newest entries in the trace buffer.

MCF51QE128 MCU Series Reference Manual, Rev. 3

18.4.1.5.13 READ_Rn

Read general-purpose CPU register   **Active Background**

0x60+CRN		Rn data [31–24]	Rn data [23–16]	Rn data [15–8]	Rn data [7–0]
host → target	D L Y	target → host	target → host	target → host	target → host
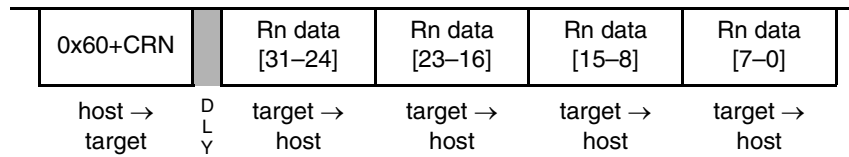
If the processor is halted, this command reads the selected CPU general-purpose register (An, Dn) and returns the 32-bit result. See Table 18-24 for the CRN details when CRG is 01.

If the processor is not halted, this command is rejected as an illegal operation and no operation is performed.

18.4.1.5.14 READ_XCSR_BYTE

Read XCSR Status Byte   **Always Available**

0x2D	XCSR [31–24]
host → target	host → target
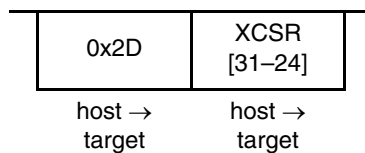
Read the special status byte of XCSR (XCSR[31–24]). This command can be executed in any mode.

18.4.1.5.15 READ_CSR2_BYTE

Read CSR2 Status Byte   **Always Available**
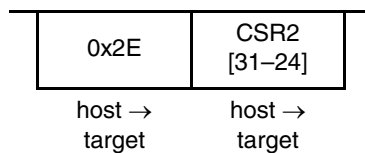
0x2E	CSR2 [31–24]
host → target	host → target

Read the most significant byte of CSR2 (CSR2[31–24]). This command can be executed in any mode.

18.4.1.5.16 READ_CSR3_BYTE

Read CSR3 Status Byte   **Always Available**

0x2F	CSR2 [31–24]
host → target	host → target
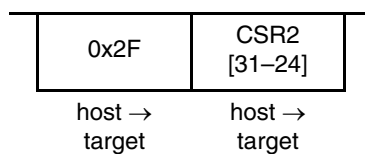
Read the most significant byte of the CSR3 (CSR3[31–24]). This command can be executed in any mode.

18.4.1.5.17 SYNC_PC

Synchronize PC to PST/DDATA Signals **Non-intrusive**

0x01	
host → target	D L Y
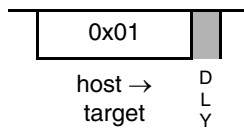
Capture the processor's current PC (program counter) and display it on the PST/DDATA signals. After the debug module receives the command, it sends a signal to the ColdFire core that the current PC must be displayed. The core responds by forcing an instruction fetch to the next PC with the address being captured by the DDATA logic. The DDATA logic captures a 2- or 3-byte instruction address, based on CSR[9]. If CSR[9] is cleared, then a 2-byte address is captured, else a 3-byte address is captured. The specific sequence of PST and DDATA values is defined as:

1. Debug signals a SYNC_PC command is pending.
2. CPU completes the current instruction.
3. CPU forces an instruction fetch to the next PC, generating a PST = 0x5 value indicating a taken branch. DDATA captures the instruction address corresponding to the PC. DDATA generates a PST marker signalling a 2- or 3-byte address as defined by CSR[9] and displays the captured PC address.

This command can be used to provide a PC synchronization point between the core's execution and the application code in the PST trace buffer. It can also be used to dynamically access the PC for performance monitoring as the execution of this command is considerably less obtrusive to the real-time operation of an application than a BACKGROUND/read-PC/GO command sequence.

18.4.1.5.18 WRITE_CREG

Write CPU control register **Active Background**

0xC0+CRN	CREG data [31–24]	CREG data [23–16]	CREG data [15–8]	CREG data [7–0]	
host → target	host → target	host → target	host → target	host → target	D L Y
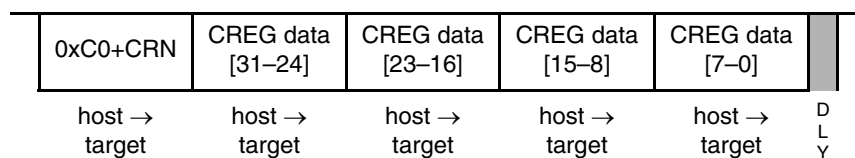
If the processor is halted, this command writes the 32-bit operand to the selected control register. This register grouping includes the PC, SR, CPUCR, VBR, and OTHER_A7. Accesses to processor control registers are always 32-bits wide, regardless of implemented register width. The register is addressed through the core register number (CRN). See Table 18-24 for the CRN details when CRG is 11.

If the processor is not halted, this command is rejected as an illegal operation and no operation is performed.

18.4.1.5.19   WRITE_DREG

Write debug control register   **Non-intrusive**

0x80+CRN	DREG data [31–24]	DREG data [23–16]	DREG data [15–8]	DREG data [7–0]	D L Y
host → target	host → target	host → target	host → target	host → target	

This command writes the 32-bit operand to the selected debug control register. This grouping includes all the debug control registers ({X}CSR*n*, BAAR, AATR, TDR, PBR*n*, PBMR, AB*x*R, DBR, DBMR). Accesses to debug control registers are always 32-bits wide, regardless of implemented register width. The register is addressed through the core register number (CRN). See Table 18-4 for CRN details.

18.4.1.5.20   WRITE_MEM.sz, WRITE_MEM.sz_WS

WRITE_MEM.sz

Write memory at the specified address   **Non-intrusive**

0x10	Address[23-0]	Memory data[7–0]	D L Y
host → target	host → target	host → target	

0x14	Address[23-0]	Memory data[15–8]	Memory data[7–0]	D L Y
host → target	host → target	host → target	host → target	

0x18	Address[23-0]	Memory data[31–24]	Memory data[23–16]	Memory data[15–8]	Memory data[7–0]	D L Y
host → target	host → target	host → target	host → target	host → target	host → target	

WRITE_MEM.sz_WS

Write memory at the specified address with status   **Non-intrusive**

0x11	Address[23-0]	Memory data[7–0]	D L Y	XCSR_SB
host → target	host → target	host → target		target → host

0x15	Address[23-0]	Memory data[15–8]	Memory data[7–0]	D L Y	XCSR_SB
host → target	host → target	host → target	host → target		target → host

0x19	Address[23-0]	Memory data[31–24]	Memory data[23–16]	Memory data[15–8]	Memory data[7–0]	D L Y	XCSR_SB
host → target	host → target	host → target	host → target	host → target	host → target		target → host

Write data at the specified memory address. The reference address is transmitted as three 8-bit packets (msb to lsb) immediately after the command packet. The access attributes are defined by BAAR[TT,TM]. The hardware forces low-order address bits to zeros for word and longword accesses to ensure these accesses are on 0-modulo-size alignments. If the with-status option is specified, the core status byte contained in XCSR[31–24] (XCSR_SB) is returned before the read data. The XCSR status byte reflects the state after the memory read was performed.

The examples show the WRITE_MEM.B{_WS}, WRITE_MEM.W{_WS}, and WRITE_MEM.L{_WS} commands.

18.4.1.5.21   WRITE_Rn

Write general-purpose CPU register   **Active Background**

0x40+CRN	Rn data [31–24]	Rn data [23–16]	Rn data [15–8]	Rn data [7–0]	D L Y
host → target	host → target	host → target	host → target	host → target	

If the processor is halted, this command writes the 32-bit operand to the selected CPU general-purpose register (An, Dn). See Table 18-24 for the CRN details when CRG is 01.

If the processor is not halted, this command is rejected as an illegal operation and no operation is performed.

18.4.1.5.22   WRITE_XCSR_BYTE

Write XCSR Status Byte   **Always Available**

0x0D	XCSR Data [31–24]
host → target	host → target
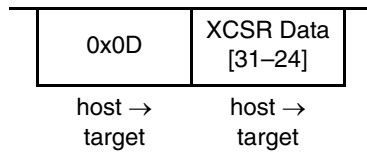
Write the special status byte of XCSR (XCSR[31–24]). This command can be executed in any mode.

18.4.1.5.23   WRITE_CSR2_BYTE

Write CSR2 Status Byte   **Always Available**
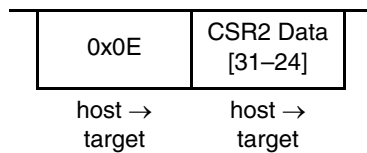
0x0E	CSR2 Data [31–24]
host → target	host → target

Write the most significant byte of CSR2 (CSR2[31–24]). This command can be executed in any mode.

18.4.1.5.24   WRITE_CSR3_BYTE

Write CSR3 Status Byte   **Always Available**

0x0F	CSR3 Data [31–24]
host → target	host → target
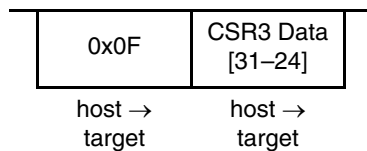
Write the most significant byte of CSR3 (CSR3[31–24]). This command can be executed in any mode.

18.4.1.6   Serial Interface Hardware Handshake Protocol

BDC commands that require CPU execution are ultimately treated at the core clock rate. Since the BDC clock source can be asynchronous relative to the bus frequency when CLKSW is cleared, it is necessary to provide a handshake protocol so the host can determine when an issued command is executed by the CPU. This section describes this protocol.

The hardware handshake protocol signals to the host controller when an issued command was successfully executed by the target. This protocol is implemented by a low pulse (16 BDC clock cycles) followed by a brief speedup pulse on the BKGD pin, generated by the target MCU when a command, issued by the host, has been successfully executed. See Figure 18-19. This pulse is referred to as the ACK pulse. After the ACK pulse is finished, the host can start the data-read portion of the command if the last-issued command was a read command, or start a new command if the last command was a write command or a control command (BACKGROUND, GO, NOP, SYNC_PC). The ACK pulse is not issued earlier than 32 BDC

clock cycles after the BDC command was issued. The end of the BDC command is assumed to be the 16th BDC clock cycle of the last bit. This minimum delay assures enough time for the host to recognize the ACK pulse. Note that there is no upper limit for the delay between the command and the related ACK pulse, since the command execution depends on the CPU bus frequency, which in some cases could be very slow compared to the serial communication rate. This protocol allows great flexibility for pod designers, since it does not rely on any accurate time measurement or short response time to any event in the serial communication.
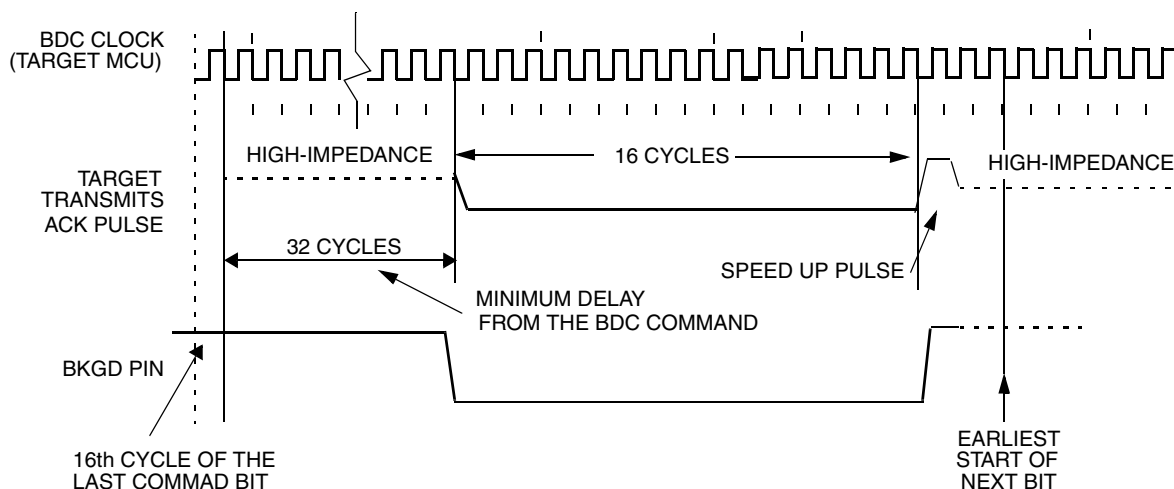
Figure 18-19. Target Acknowledge Pulse (ACK)

NOTE

If the ACK pulse was issued by the target, the host assumes the previous command was executed. If the CPU enters a stop mode prior to executing a non-intrusive command, the command is discarded and the ACK pulse is not issued. After entering a stop mode, the BDC command is no longer pending and the XCSR[CSTAT] value of 001 is kept until the next command is successfully executed.

Figure 18-20 shows the ACK handshake protocol in a command level timing diagram. A READ_MEM.B command is used as an example:

1. The 8-bit command code is sent by the host, followed by the address of the memory location to be read.
2. The target BDC decodes the command and sends it to the CPU.
3. Upon receiving the BDC command request, the CPU schedules a execution slot for the command.
4. The CPU temporarily stalls the instruction stream at the scheduled point, executes the READ_BYTE command and then continues.

This process is referred to as cycle stealing. The READ_MEM.B appears as a single-cycle operation to the processor, even though the pipelined nature of the Operand Execution Pipeline requires multiple CPU clock cycles for it to actually complete. After that, the debug module tracks the execution of the READ_MEM.b command as the processor resumes the normal flow of the application program. After detecting the READ_MEM.B command is done, the BDC issues an ACK pulse to the host controller,

---

MCF51QE128 MCU Series Reference Manual, Rev. 3

indicating that the addressed byte is ready to be retrieved. After detecting the ACK pulse, the host initiates the data-read portion of the command.
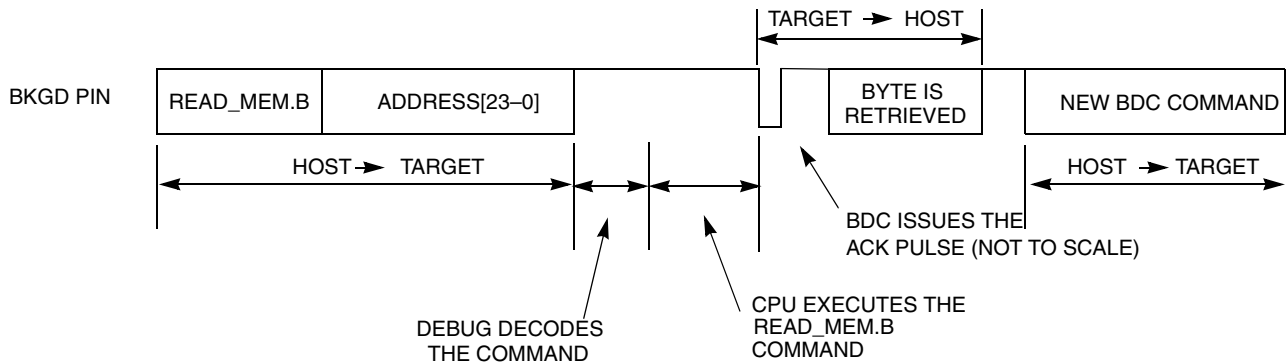

Figure 18-20. Handshake Protocol at Command Level

Unlike a normal bit transfer, where the host initiates the transmission by issuing a negative edge in the BKGD pin, the serial interface ACK handshake pulse is initiated by the target MCU. The hardware handshake protocol in Figure 18-20 specifies the timing when the BKGD pin is being driven, so the host should follow these timing constraints in order to avoid the risks of an electrical conflict at the BKGD pin.

The ACK handshake protocol does not support nested ACK pulses. If a BDC command is not acknowledged by an ACK pulse, the host first needs to abort the pending command before issuing a new BDC command. When the CPU enters a stop mode at about the same time the host issues a command that requires CPU execution, the target discards the incoming command. Therefore, the command is not acknowledged by the target, meaning that the ACK pulse is not issued in this case. After a certain time, the host could decide to abort the ACK protocol in order allow a new command. Therefore, the protocol provides a mechanism where a command (a pending ACK) could be aborted. Unlike a regular BDC command, the ACK pulse does not provide a timeout. In the case of a STOP instruction where the ACK is prevented from being issued, it would remain pending indefinitely if not aborted. See the handshake abort procedure described in Section 18.4.1.7, "Hardware Handshake Abort Procedure."

18.4.1.7   Hardware Handshake Abort Procedure

The abort procedure is based on the SYNC command. In order to abort a command that has not responded with an ACK pulse, the host controller generates a sync request (by driving BKGD low for at least 128 serial clock cycles and then driving it high for one serial clock cycle as a speedup pulse). By detecting this long low pulse on the BKGD pin, the target executes the sync protocol (see Section 18.4.1.5.1, "SYNC"), and assumes that the pending command and therefore the related ACK pulse, are being aborted. Therefore, after the sync protocol completes, the host is free to issue new BDC commands.

Note, since the host knows the target BDC clock frequency, the SYNC command does not need to consider the lowest possible target frequency. In this case, the host could issue a SYNC very close to the 128 serial clock cycles length, just providing a small overhead on the pulse length in order to assure the sync pulse is not misinterpreted by the target.

It is important to notice that any issued BDC command that requires CPU execution is scheduled for execution by the pipeline based on the dynamic state of the machine, provided the processor does not enter

any of the stop modes. If the host aborts a command by sending the sync pulse, it should then read XCSR[CSTAT] after the sync response is issued by the target, checking for CSTAT cleared, before attempting to send any new command that requires CPU execution. This prevents the new command from being discarded at the debug/CPU interface, due to the pending command being executed by the CPU. Any new command should be issued only after XCSR[CSTAT] is cleared.

There are multiple reasons that could cause a command to take too long to execute, measured in terms of the serial communication rate. Either the BDC clock frequency is much faster than the CPU clock frequency, or the CPU is accessing a slow memory, which would cause pipeline stall cycles to occur. All commands referencing the CPU registers or memory require access to the processor's local bus to complete. If the processor is executing a tight loop that is contained within a single aligned longword, the processor may never successfully grant the internal bus to the debug command. For example:

```
        align   4
label1: nop
        bra.b   label1
or

        align   4
label2: bra.w   label2
```

These two examples of tight loops both exhibit the BDM lockout behavior. If the loop spans across two longwords, there are no issues, so the recommended construct is:

```
        align   4
label3: bra.l   label3
```

The hardware handshake protocol is appropriate for these situations, but the host could also decide to use the software handshake protocol instead. In this case, if XCSR[CSTAT] is 001, there is a BDC command pending at the debug/CPU interface. The host controller should monitor XCSR[CSTAT] and wait until it is 000 in order to be able to issue a new command that requires CPU execution. However, if the XCSR[CSTAT] is 1*xx*, the host should assume the last command failed to execute. To recover from this condition, the following sequence is suggested:

1. Issue a SYNC command to reset the BDC communication channel.
2. The host issues a BDM NOP command.
3. The host reads the channel status using a READ_XCSR_BYTE command.
4. If XCSR[CSTAT] is 000
    then the status is okay; proceed
    else
        Halt the CPU using a BDM BACKGROUND command
        Repeat steps 1,2,3
        If XCSR[CSTAT] is 000, then proceed, else reset the device

Figure 18-21 shows a SYNC command aborting a READ_BYTE. Note that after the command is aborted, a new command could be issued by the host.

NOTE

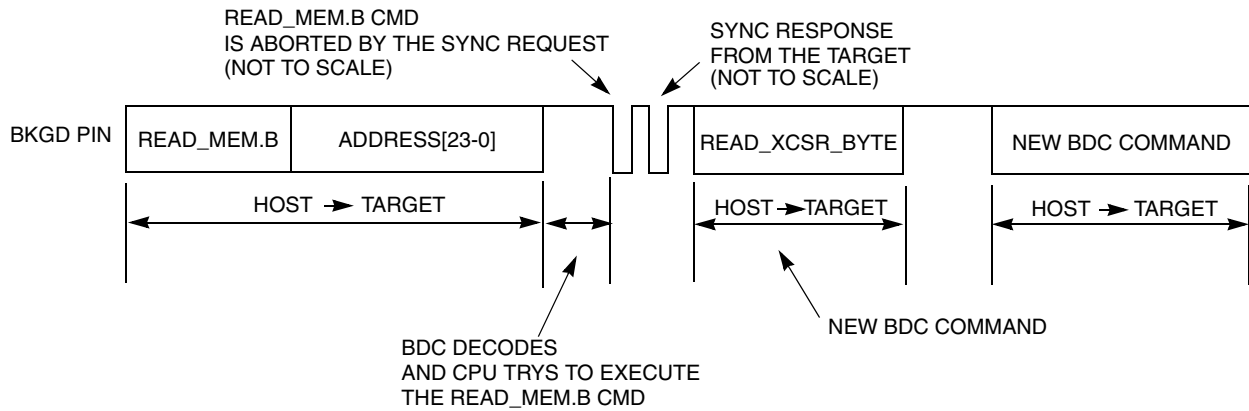Figure 18-21 signal timing is not drawn to scale.

Figure 18-21. ACK Abort Procedure at the Command Level

Figure 18-22 a shows a conflict between the ACK pulse and the sync request pulse. This conflict could occur if a pod device is connected to the target BKGD pin and the target is already executing a BDC command. Consider that the target CPU is executing a pending BDC command at the exact moment the pod is being connected to the BKGD pin. In this case, an ACK pulse is issued at the same time as the SYNC command. In this case there is an electrical conflict between the ACK speedup pulse and the sync pulse. Since this is not a probable situation, the protocol does not prevent this conflict from happening.
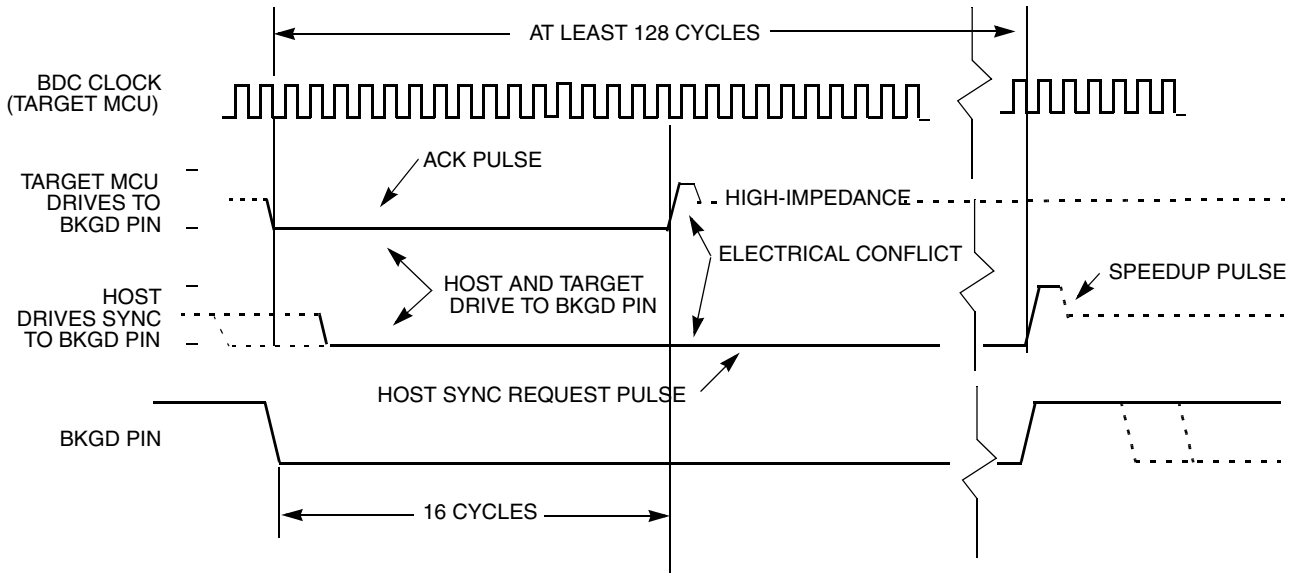
Figure 18-22. ACK Pulse and SYNC Request Conflict

The hardware handshake protocol is enabled by the ACK_ENABLE command and disabled by the ACK_DISABLE command. It also allows for pod devices to choose between the hardware handshake protocol or the software protocol that monitors the XCSR status byte. The ACK_ENABLE and ACK_DISABLE commands are:

- ACK_ENABLE — Enables the hardware handshake protocol. The target issues the ACK pulse when a CPU command is executed. The ACK_ENABLE command itself also has the ACK pulse as a response.

- ACK_DISABLE — Disables the ACK pulse protocol. In this case, the host should verify the state of XCSR[CSTAT] in order to evaluate if there are pending commands and to check if the CPU's operating state has changed to or from active background mode via XCSR[31–30].

The default state of the protocol, after reset, is hardware handshake protocol disabled.

The commands that do not require CPU execution, or that have the status register included in the retrieved bit stream, do not perform the hardware handshake protocol. Therefore, the target does not respond with an ACK pulse for those commands even if the hardware protocol is enabled. Conversely, only commands that require CPU execution and do not include the status byte perform the hardware handshake protocol. See the "ACK if Enabled?" column in Table 18-25 for the complete enumeration of this function.

An exception is the ACK_ENABLE command, which does not require CPU execution but responds with the ACK pulse. This feature can be used by the host to evaluate if the target supports the hardware handshake protocol. If an ACK pulse is issued in response to this command, the host knows that the target supports the hardware handshake protocol. If the target does not support the hardware handshake protocol the ACK pulse is not issued. In this case, the ACK_ENABLE command is ignored by the target, since it is not recognized as a valid command.

18.4.2   Real-Time Debug Support

The ColdFire family supports debugging real-time applications. For these types of embedded systems, the processor must continue to operate during debug. The foundation of this area of debug support is that while the processor cannot be halted to allow debugging, the system can generally tolerate the small intrusions with minimal effect on real-time operation.

NOTE

The details regarding real-time debug support will be supplied at a later time.

18.4.3   Real-Time Trace Support

The classic ColdFire debug architecture supports real-time trace via the PST/DDATA output signals. For this functionality, the following apply:

- One (or more) PST value is generated for each executed instruction
- Branch target instruction address information is displayed on all non-PC-relative change-of-flow instructions, where the user selects a programmable number of bytes of target address
  - Displayed information includes PST marker plus target instruction address as DDATA
  - Captured address creates the appropriate number of DDATA entries, each with 4 bits of address
- Optional data trace capabilities are provided for accesses mapped to the slave peripheral bus
  - Displayed information includes PST marker plus captured operand value as DDATA
  - Captured operand creates the appropriate number of DDATA entries, each with 4 bits of data

The resulting PST/DDATA output stream, with the application program memory image, provides an instruction-by-instruction dynamic trace of the execution path.

For the V1 ColdFire core and its single debug signal, support for trace functionality is completely redefined. The V1 solution provides an on-chip PST/DDATA trace buffer (known as the PSTB) to record the stream of PST and DDATA values.

Even with the application of a PST trace buffer, problems associated with the PST bandwidth and associated fill rate of the buffer remain. Given that there is one (or more) PST entry per instruction, the PSTB would fill rapidly without some type of data compression. Luckily, the PST compression technology was previously developed and included as part of the Version 5 ColdFire core (although very different than the resulting V1 implementation).

Consider the following example to illustrate the PST compression algorithm. Most sequential instructions generate a single PST = 1 value. Without compression, the execution of ten sequential instructions generates a stream of ten PST = 1 values. With PST compression, the reporting of any PST = 1 value is delayed so that consecutive PST = 1 values can be accumulated. When a PST ≠ 1 value is reported, the maximum accumulation count reached, or a debug data value captured, a single accumulated PST value is generated. Returning to the example with compression enabled, the execution of ten sequential instructions generates a single PST value indicating ten sequential instructions have been executed.

This technique has proven to be effective at significantly reducing the average PST entries per instruction and PST entries per machine cycle. The application of this compression technique makes the application of a useful PST trace buffer for the V1 ColdFire core realizable. The resulting 5-bit PST definitions are shown in Table 18-26.

Table 18-26. Processor Status Encodings

PST[4:0]	Definition
0x00	Continue execution. Many instructions execute in one processor cycle. If an instruction requires more processor clock cycles, subsequent clock cycles are indicated by driving PST with this encoding.
0x01	Begin execution of one instruction. For most instructions, this encoding signals the first processor clock cycle of an instruction's execution. Certain change-of-flow opcodes, plus the PULSE and WDDATA instructions, generate different encodings.
0x02	Reserved
0x03	Entry into user-mode. Signaled after execution of the instruction that caused the ColdFire processor to enter user mode.
0x04	Begin execution of PULSE and WDDATA instructions. PULSE defines triggers or markers for debug and/or performance analysis. WDDATA lets the core write any operand (byte, word, or longword) directly to the DDATA port, independent of debug module configuration. When WDDATA is executed, a value of 0x04 is signaled on the PST port, followed by the appropriate marker, and then the data transfer on the DDATA port. The number of captured data bytes depends on the WDDATA operand size.
0x05	Begin execution of taken branch or SYNC_PC BDM command. For some opcodes, a branch target address may be displayed on DDATA depending on the CSR settings. CSR also controls the number of address bytes displayed, indicated by the PST marker value preceding the DDATA nibble that begins the data output. This encoding also indicates that the SYNC_PC command has been processed.
0x06	Reserved
0x07	Begin execution of return from exception (RTE) instruction.

Table 18-26. Processor Status Encodings (continued)

PST[4:0]	Definition
0x08–0x0B	Indicates the number of data bytes to be displayed as DDATA on subsequent processor clock cycles. This marker value is driven as the PST one processor clock cycle before the data is displayed on DDATA. The capturing of peripheral bus data references is controlled by CSR[DDC]. 0x08 Begin 1-byte data transfer on DDATA 0x09 Begin 2-byte data transfer on DDATA 0x0A Reserved 0x0B Begin 4-byte data transfer on DDATA
0x0C–0x0F	Indicates the number of address bytes to be displayed as DDATA on subsequent processor clock cycles. This marker value is driven as the PST one processor clock cycle before the address is displayed on DDATA. The capturing of branch target addresses is controlled by CSR[BTB]. 0x0C Reserved 0x0D Begin 2-byte address transfer on DDATA (Displayed address is shifted right 1: ADDR[17:1]) 0x0E Begin 3-byte address transfer on DDATA (Displayed address is shifted right 1: ADDR[24:1]) 0x0F Reserved
0x10–0x11	Reserved
0x12	Completed execution of 2 sequential instructions
0x13	Completed execution of 3 sequential instructions
0x14	Completed execution of 4 sequential instructions
0x15	Completed execution of 5 sequential instructions
0x16	Completed execution of 6 sequential instructions
0x17	Completed execution of 7 sequential instructions
0x18	Completed execution of 8 sequential instructions
0x19	Completed execution of 9 sequential instructions
0x1A	Completed execution of 10 sequential instructions
0x1B	This value signals there has been a change in the breakpoint trigger state machine. It appears as a single marker for each state change and is immediately followed by a DDATA value signaling the new breakpoint trigger state encoding.
0x1C	Exception processing. This value signals the processor has encountered an exception condition. Although this is a multi-cycle mode, there are only two PST = 0x1C values recorded before the mode value is suppressed.
0x1D	Emulator mode exception processing. This value signals the processor has encountered a debug interrupt or a properly-configured trace exception. Although this is a multi-cycle mode, there are only two PST = 0x1D values recorded before the mode value is suppressed.
0x1E	Processor is stopped. This value signals the processor has executed a STOP instruction. Although this is a multi-cycle mode since the ColdFire processor remains stopped until an interrupt or reset occurs, there are only two PST = 0x1E values recorded before the mode value is suppressed.
0x1F	Processor is halted. This value signals the processor has been halted. Although this is a multi-cycle mode since the ColdFire processor remains halted until a BDM go command is received or reset occurs, there are only two PST = 0x1F values recorded before the mode value is suppressed.

18.4.3.1 Begin Execution of Taken Branch (PST = 0x05)

PST is 0x5 when a taken branch is executed. For some opcodes, a branch target address may be loaded into the trace buffer (PSTB) depending on the CSR settings. CSR also controls the number of address bytes

loaded, which is indicated by the PST marker value immediately preceding the DDATA entry in the PSTB that begins the address entries.

Multiple byte DDATA values are displayed in least-to-most-significant order. The processor captures only those target addresses associated with taken branches that use a variant addressing mode (RTE and RTS instructions and JMP and JSR instructions using address register indirect or indexed addressing modes, and all exceptio vectors).

The simplest example of a branch instruction using a variant address is the compiled code for a C language case statement. Typically, the evaluation of this statement uses the variable of an expression as an index into a table of offsets, where each offset points to a unique case within the structure. For such change-of-flow operations, the ColdFire processor loads the PSTB as follows:

1. Load PST=0x05 to identify that a taken branch is executed.
2. Optionally load the marker for the target address capture. Encodings 0x0D or 0x0E identify the number of bytes loaded into the PSTB.
3. The new target address is optionally available into the PSTB. The number of bytes of the target address loaded is configurable (2 or 3 bytes, where the encoding is 0x0D and 0x0E, respectively).

Another example of a variant branch instruction would be a JMP (A0) instruction. Figure 18-23 shows the PSTB entries that indicate a JMP (A0) execution, assuming the CSR was programmed to display the lower 2 bytes of an address.

PST Values	Description
0x05	Taken Branch
0x0D	2-byte Address
{10, Address[3:0]}	
{10, Address[7:4]}	
{10, Address[11:8]}	
{10, Address[15:12]}	

Figure 18-23. Example JMP Instruction Output in PSTB

PST of 0x05 indicates a taken branch and the marker value 0x0D indicates a 2-byte address. Thus, the following entries display the lower two bytes of address register A0 in least-to-most-significant nibble order. The next PST entry after the JMP instruction completes depends on the target instruction. See Section 18.4.3.2, "PST Trace Buffer (PSTB)," for entry descriptions explaining the 2-bit prefix before each address nibble.

18.4.3.2 PST Trace Buffer (PSTB)

As PST and DDATA values are captured and loaded in the trace buffer, each entry is 6 bits in size so the type of the entry can easily be determined when post-processing the PSTB. See Figure 18-24.
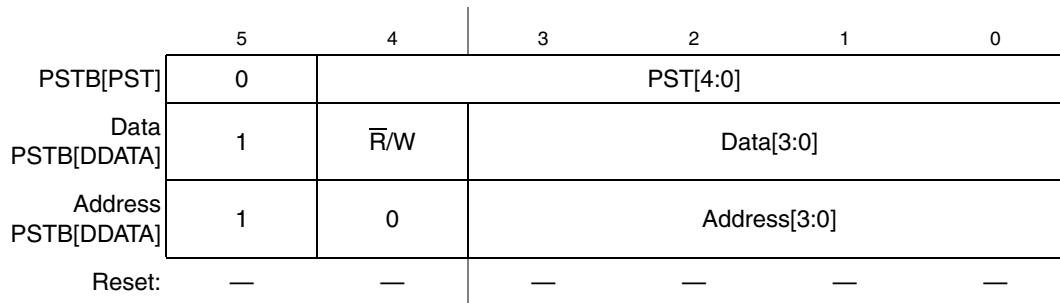
	5	4	3	2	1	0
PSTB[PST]	0	PST[4:0]				
Data PSTB[DDATA]	1	$\overline{\text{R}}$/W	Data[3:0]			
Address PSTB[DDATA]	1	0	Address[3:0]			
Reset:	—	—	—	—	—	—

Figure 18-24. V1 PST/DDATA Trace Buffer Entry Format

18.4.3.3 PST/DDATA Example

In this section, an example showing the behavior of the PST/DDATA functionality is detailed. Consider the following interrupt service routine that counts the interrupt, then negates the IRQ, and performs a software IACK and then exits. This example is presented here because it exercises a considerable set of the PST/DDATA capabilities.

```
                       _isr:
01074: 46fc 2700        mov.w   &0x2700,%sr        # disable interrupts
01078: 2f08             mov.l   %a0,-(%sp)         # save a0
0107a: 2f00             mov.l   %d0,-(%sp)         # save d0
0107c: 302f 0008        mov.w   (8,%sp),%d0        # load format/vector word
01080: e488             lsr.l   &2,%d0             # align vector number
01082: 0280 0000 00ff   andi.l  &0xff,%d0          # isolate vector number
01088: 207c 0080 1400   mov.l   &int_count,%a0     # base of interrupt counters

                       _isr_entry1:
0108e: 52b0 0c00        addq.l  &1,(0,%a0,%d0.l*4) # count the interrupt
01092: 11c0 a021        mov.b   %d0,IGCR0+1.w      # negate the irq
01096: 1038 a020        mov.b   IGCR0.w,%d0        # force the write to complete
0109a: 4e71             nop                        # synchronize the pipelines
0109c: 71b8 ffe0        mvz.b   SWIACK.w,%d0       # software iack: pending irq?
010a0: 0c80 0000 0041   cmpi.l  %d0,&0x41          # level 7 or none pending?
010a6: 6f08             ble.b   _isr_exit          # yes, then exit
010a8: 52b9 0080 145c   addq.l  &1,swiack_count    # increment the swiack count
010ae: 60de             bra.b   _isr_entry1        # continue at entry1

                       _isr_exit:
010b0: 201f             mov.l   (%sp)+,%d0         # restore d0
010b2: 205f             mov.l   (%sp)+,%a0         # restore a0
010b4: 4e73             rte                        # exit
```

This ISR executes mostly as straight-line code: there is a single conditional branch @ PC = 0x10A6, which is taken in this example. The following description includes the PST and DDATA values generated as this code snippet executes. In this example, the CSR setting enables the display of 2-byte branch addresses. Peripheral bus read and write operands are being traced. The sequence begins with an interrupt exception:

```
interrupt exception occurs @ pc = 5432 while in user mode
```

```
                                                   # pst   = 1c, 1c, 05, 0d
                                                   # ddata = 2a, 23, 28, 20
                                                   #        trg_addr = 083a << 1
                                                   #        trg_addr = 1074
                    _isr:
01074: 46fc 2700        mov.w  &0x2700,%sr         # pst   = 01
01078: 2f08             mov.l  %a0,-(%sp)          # pst   = 01
0107a: 2f00             mov.l  %d0,-(%sp)          # pst   = 01
0107c: 302f 0008        mov.w  (8,%sp),%d0         # pst   = 01
01080: e488             lsr.l  &2,%d0              # pst   = 01
01082: 0280 0000 00ff   andi.l &0xff,%d0           # pst   = 01
01088: 207c 0080 1400   mov.l  &int_count,%a0      # pst   = 01
0108e: 52b0 0c00        addq.l &1,(0,%a0,%d0.l*4)  # pst   = 01
01092: 11c0 a021        mov.b  %d0,IGCR0+1.w       # pst   = 01, 08
                                                   # ddata = 30, 30
                                                   #        wdata.b = 0x00
01096: 1038 a020        mov.b  IGCR0.w,%d0         # pst   = 01, 08
                                                   # ddata = 28, 21
                                                   #        rdata.b = 0x18
0109a: 4e71             nop                        # pst   = 01
0109c: 71b8 ffe0        mvz.b  SWIACK.w,%d0        # pst   = 01, 08
                                                   # ddata = 20, 20
                                                   #        rdata.b = 0x00
010a0: 0c80 0000 0041   cmpi.l %d0,&0x41           # pst   = 01
010a6: 6f08             ble.b  _isr_exit           # pst   = 05 (taken branch)
010b0: 201f             mov.l  (%sp)+,%d0          # pst   = 01
010b2: 205f             mov.l  (%sp)+,%a0          # pst   = 01
010b4: 4e73             rte                        # pst   = 07, 03, 05, 0d
                                                   # ddata = 29, 21, 2a, 22
                                                   #trg_addr = 2a19 << 1
                                                   #trg_addr = 5432
```

As the PSTs are compressed, the resulting stream of 6-bit hexadecimal entries is loaded into consecutive locations in the PST trace buffer:

```
PSTB[*]= 1c, 1c, 05, 0d,   // interrupt exception
         2a, 23, 28, 20,   // branch target addr = 1074
         19, 08, 30, 30,   // 9 sequential insts, write byte
         01, 08, 28, 21,   // 1 sequential inst,  read byte
         12, 08, 20, 20,   // 2 sequential insts, read byte
         01, 05, 12,       // 1 + taken_branch + 2 sequential
         07, 03, 05, 0d,   // rte, entry into user mode
         29, 21, 2a, 22    // branch target addr = 5432
```

Architectural studies on the compression algorithm determined an appropriate size for the PST trace buffer. Using a suite of ten MCU benchmarks, a 64-entry PSTB was found to capture an average window of time of 520 processor cycles with program trace using 2-byte addresses enabled.

18.4.3.4   Processor Status, Debug Data Definition

This section specifies the ColdFire processor and debug module's generation of the processor status (PST) and debug data (DDATA) output on an instruction basis. In general, the PST/DDATA output for an instruction is defined as follows:

PST = 0x01, {PST = [0x89B], DDATA = operand}

where the {...} definition is optional operand information defined by the setting of the CSR.

The CSR provides capabilities to display operands based on reference type (read, write, or both). A PST value {0x08, 0x09, or 0x0B} identifies the size and presence of valid data to follow in the PST trace buffer (PSTB) {1, 2, or 4 bytes}. Additionally, for certain change-of-flow instructions, CSR[BTB] provides the capability to display the target instruction address in the PSTB {2 or 3 bytes} using a PST value of {0x0D or 0x0E}.

18.4.3.4.1   User Instruction Set

Table 18-27 shows the PST/DDATA specification for user-mode instructions. Rn represents any {Dn, An} register. In this definition, the y suffix generally denotes the source, and x denotes the destination operand. For a given instruction, the optional operand data is displayed only for those effective addresses referencing memory. The DD nomenclature refers to the DDATA outputs.

Table 18-27. PST/DDATA Specification for User-Mode Instructions

Instruction	Operand Syntax	PST/DDATA
add.l	<ea>y,Dx	PST = 0x01, {PST = 0x0B, DD = source operand}
add.l	Dy,<ea>x	PST = 0x01, {PST = 0x0B, DD = source}, {PST = 0x0B, DD = destination}
adda.l	<ea>y,Ax	PST = 0x01, {PST = 0x0B, DD = source operand}
addi.l	#<data>,Dx	PST = 0x01
addq.l	#<data>,<ea>x	PST = 0x01, {PST = 0x0B, DD = source}, {PST = 0x0B, DD = destination}
addx.l	Dy,Dx	PST = 0x01
and.l	<ea>y,Dx	PST = 0x01, {PST = 0x0B, DD = source operand}
and.l	Dy,<ea>x	PST = 0x01, {PST = 0x0B, DD = source}, {PST = 0x0B, DD = destination}
andi.l	#<data>,Dx	PST = 0x01
asl.l	{Dy,#<data>},Dx	PST = 0x01
asr.l	{Dy,#<data>},Dx	PST = 0x01
bcc.{b,w,l}		if taken, then PST = 0x05, else PST = 0x01
bchg.{b,l}	#<data>,<ea>x	PST = 0x01, {PST = 0x08, DD = source}, {PST = 0x08, DD = destination}
bchg.{b,l}	Dy,<ea>x	PST = 0x01, {PST = 0x08, DD = source}, {PST = 0x08, DD = destination}
bclr.{b,l}	#<data>,<ea>x	PST = 0x01, {PST = 0x08, DD = source}, {PST = 0x08, DD = destination}
bclr.{b,l}	Dy,<ea>x	PST = 0x01, {PST = 0x08, DD = source}, {PST = 0x08, DD = destination}
bitrev.l	Dx	PST = 0x01
bra.{b,w,l}		PST = 0x05
bset.{b,l}	#<data>,<ea>x	PST = 0x01, {PST = 0x08, DD = source}, {PST = 0x08, DD = destination}
bset.{b,l}	Dy,<ea>x	PST = 0x01, {PST = 0x08, DD = source}, {PST = 0x08, DD = destination}
bsr.{b,w,l}		PST = 0x05, {PST = 0x0B, DD = destination operand}
btst.{b,l}	#<data>,<ea>x	PST = 0x01, {PST = 0x08, DD = source operand}

Table 18-27. PST/DDATA Specification for User-Mode Instructions (continued)

Instruction	Operand Syntax	PST/DDATA
btst.{b,l}	Dy,<ea>x	PST = 0x01, {PST = 0x08, DD = source operand}
byterev.l	Dx	PST = 0x01
clr.b	<ea>x	PST = 0x01, {PST = 0x08, DD = destination operand}
clr.l	<ea>x	PST = 0x01, {PST = 0x0B, DD = destination operand}
clr.w	<ea>x	PST = 0x01, {PST = 0x09, DD = destination operand}
cmp.b	<ea>y,Dx	PST = 0x01, {0x08, source operand}
cmp.l	<ea>y,Dx	PST = 0x01, {PST = 0x0B, DD = source operand}
cmp.w	<ea>y,Dx	PST = 0x01, {0x09, source operand}
cmpa.l	<ea>y,Ax	PST = 0x01, {PST = 0x0B, DD = source operand}
cmpa.w	<ea>y,Ax	PST = 0x01, {0x09, source operand}
cmpi.b	#<data>,Dx	PST = 0x01
cmpi.l	#<data>,Dx	PST = 0x01
cmpi.w	#<data>,Dx	PST = 0x01
eor.l	Dy,<ea>x	PST = 0x01, {PST = 0x0B, DD = source}, {PST = 0x0B, DD = destination}
eori.l	#<data>,Dx	PST = 0x01
ext.l	Dx	PST = 0x01
ext.w	Dx	PST = 0x01
extb.l	Dx	PST = 0x01
illegal		PST = 0x01[1]
jmp	<ea>y	PST = 0x05, {PST = [0x0{DE}], DD = target address}[2]
jsr	<ea>y	PST = 0x05, {PST = [0x0{DE}], DD = target address}, {PST = 0x0B, DD = destination operand}[2]
lea.l	<ea>y,Ax	PST = 0x01
link.w	Ay,#<displacement>	PST = 0x01, {PST = 0x0B, DD = destination operand}
lsl.l	{Dy,#<data>},Dx	PST = 0x01
lsr.l	{Dy,#<data>},Dx	PST = 0x01
mov3q.l	#<data>,<ea>x	PST = 0x01, {PST = 0x0B,DD = destination operand}
move.b	<ea>y,<ea>x	PST = 0x01, {PST = 0x08, DD = source}, {PST = 0x08, DD = destination}
move.l	<ea>y,<ea>x	PST = 0x01, {PST = 0x0B, DD = source}, {PST = 0x0B, DD = destination}
move.w	<ea>y,<ea>x	PST = 0x01, {PST = 0x09, DD = source}, {PST = 0x09, DD = destination}
move.w	CCR,Dx	PST = 0x01
move.w	{Dy,#<data>},CCR	PST = 0x01
movea.l	<ea>y,Ax	PST = 0x01, {PST = 0x0B, DD = source}

Table 18-27. PST/DDATA Specification for User-Mode Instructions (continued)

Instruction	Operand Syntax	PST/DDATA
movea.w	<ea>y,Ax	PST = 0x01, {PST = 0x09, DD = source}
movem.l	#list,<ea>x	PST = 0x01, {PST = 0x0B, DD = destination},...
movem.l	<ea>y,#list	PST = 0x01, {PST = 0x0B, DD = source},...
moveq.l	#<data>,Dx	PST = 0x01
muls.l	<ea>y,Dx	PST = 0x01, {PST = 0x0B, DD = source operand}
muls.w	<ea>y,Dx	PST = 0x01, {PST = 0x09, DD = source operand}
mulu.l	<ea>y,Dx	PST = 0x01, {PST = 0x0B, DD = source operand}
mulu.w	<ea>y,Dx	PST = 0x01, {PST = 0x09, DD = source operand}
mvs.b	<ea>y,Dx	PST = 0x01, {0x08, source operand}
mvs.w	<ea>y,Dx	PST = 0x01, {0x09, source operand}
mvz.b	<ea>y,Dx	PST = 0x01, {0x08, source operand}
mvz.w	<ea>y,Dx	PST = 0x01, {0x09, source operand}
neg.l	Dx	PST = 0x01
negx.l	Dx	PST = 0x01
nop		PST = 0x01
not.l	Dx	PST = 0x01
or.l	<ea>y,Dx	PST = 0x01, {PST = 0x0B, DD = source operand}
or.l	Dy,<ea>x	PST = 0x01, {PST = 0x0B, DD = source}, {PST = 0x0B, DD = destination}
ori.l	#<data>,Dx	PST = 0x01
pea.l	<ea>y	PST = 0x01, {PST = 0x0B, DD = destination operand}
pulse		PST = 0x04
rts		PST = 0x01, {PST = 0x0B, DD = source operand}, PST = 0x05, {PST = [0x0{DE}], DD = target address}
sats.l	Dx	PST = 0x01
scc.b	Dx	PST = 0x01
sub.l	<ea>y,Dx	PST = 0x01, {PST = 0x0B, DD = source operand}
sub.l	Dy,<ea>x	PST = 0x01, {PST = 0x0B, DD = source}, {PST = 0x0B, DD = destination}
suba.l	<ea>y,Ax	PST = 0x01, {PST = 0x0B, DD = source operand}
subi.l	#<data>,Dx	PST = 0x01
subq.l	#<data>,<ea>x	PST = 0x01, {PST = 0x0B, DD = source}, {PST = 0x0B, DD = destination}
subx.l	Dy,Dx	PST = 0x01
swap.w	Dx	PST = 0x01
tas.b	<ea>x	PST = 0x01, {0x08, source}, {0x08, destination}

MCF51QE128 MCU Series Reference Manual, Rev. 3

Table 18-27. PST/DDATA Specification for User-Mode Instructions (continued)

Instruction	Operand Syntax	PST/DDATA
tpf		PST = 0x01
tpf.l	#<data>	PST = 0x01
tpf.w	#<data>	PST = 0x01
trap	#<data>	PST = 0x01[1]
tst.b	<ea>x	PST = 0x01, {PST = 0x08, DD = source operand}
tst.l	<ea>y	PST = 0x01, {PST = 0x0B, DD = source operand}
tst.w	<ea>y	PST = 0x01, {PST = 0x09, DD = source operand}
unlk	Ax	PST = 0x01, {PST = 0x0B, DD = destination operand}
wddata.b	<ea>y	PST = 0x04, PST = 0x08, DD = source operand
wddata.l	<ea>y	PST = 0x04, PST = 0x0B, DD = source operand
wddata.w	<ea>y	PST = 0x04, PST = 0x09, DD = source operand

[1] During normal exception processing, the PSTB is loaded with two successive 0x1C entries indicating the exception processing state. The exception stack write operands, as well as the vector read and target address of the exception handler may also be displayed.

```
Exception Processing:
     PST = 0x1C, 0x1C,
    {PST = 0x0B,DD = destination},      // stack frame
    {PST = 0x0B,DD = destination},      // stack frame
    {PST = 0x0B,DD = source},           // vector read
     PST = 0x05,{PST = [0x0{DE}],DD = target}// handler PC
```

A similar set of PST/DD values is generated in response to an emulator mode exception. For these events (caused by a debug interrupt or properly-enabled trace exception), the initial PST values are 0x1D, 0x1D and the remaining sequence is equivalent to normal exception processing.

The PST/DDATA specification for the reset exception is shown below:

```
Exception Processing:
     PST = 0x1C, 0x1C,
     PST = 0x05,{PST = [0x0{DE}],DD = target}//  handler PC
```

The initial references at address 0 and 4 are never captured nor displayed since these accesses are treated as instruction fetches.

For all types of exception processing, the PST = 0x1C (or 0x1D) value is driven for two trace buffer entries.

[2] For JMP and JSR instructions, the optional target instruction address is displayed only for those effective address fields defining variant addressing modes. This includes the following <ea>x values: (An), (d16,An), (d8,An,Xi), (d8,PC,Xi).

18.4.3.4.2   Supervisor Instruction Set

The supervisor instruction set has complete access to the user mode instructions plus the opcodes shown below. The PST/DDATA specification for these opcodes is shown in Table 18-28.

Table 18-28. PST/DDATA Specification for Supervisor-Mode Instructions

Instruction	Operand Syntax	PST/DDATA
halt		PST = 0x01, PST = 0x0F
move.l	Ay,USP	PST = 0x01
move.l	USP,Ax	PST = 0x01
move.w	SR,Dx	PST = 0x01
move.w	{Dy,#<data>},SR	PST = 0x01, {PST = 0x03}
movec.l	Ry,Rc	PST = 0x01
rte		PST = 0x07, {PST = 0x0B, DD = source operand}, {PST = 0x03}, {PST = 0x0B, DD = source operand}, PST = 0x05, {[PST = 0x0{DE}], DD = target address}
stldsr.w	#imm	PST = 0x01, {PST = 0x0A, DD = destination operand, PST = 0x03}
stop	#<data>	PST = 0x01, PST = 0x1E
wdebug.l	<ea>y	PST = 0x01, {PST = 0x0B, DD = source, PST = 0x0B, DD = source}

The move-to-SR, STLDSR, and RTE instructions include an optional PST = 0x3 value, indicating an entry into user mode.

Similar to the exception processing mode, the stopped state (PST = 0x1E) and the halted state (PST = 0x1F) display this status for two entries when the ColdFire processor enters the given mode.

18.4.4 Freescale-Recommended BDM Pinout

Typically, a relatively simple interface pod is used to translate commands from a host computer into commands for the custom serial interface to the single-wire background debug system. Depending on the development tool vendor, this interface pod may use a standard RS-232 serial port, a parallel printer port, or some other type of communications such as a universal serial bus (USB) to communicate between the host PC and the pod. The pod typically connects to the target system with ground, the BKGD pin, $\overline{\text{RESET}}$, and sometimes V_{DD}. An open-drain connection to reset allows the host to force a target system reset, useful to regain control of a lost target system or to control startup of a target system before the on-chip nonvolatile memory has been programmed. Sometimes V_{DD} can be used to allow the pod to use power from the target system to avoid the need for a separate power supply. However, if the pod is powered separately, it can be connected to a running target system without forcing a target system reset or otherwise disturbing the running application program.

```
BKGD        1 | ■  ● | 2  GND
No Connect  3 | ●  ● | 4  RESET
No Connect  5 | ●  ● | 6  VDD
```
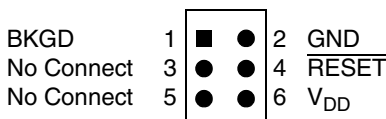
Figure 25. Recommended BDM Connector

Appendix A
Revision History

This appendix describes corrections to the *MCF51QE128 Reference Manual*. For convenience, the corrections are grouped by revision.

A.1   Changes between Rev. 2 and Rev. 3

Table 29. MCF51QE128RM Rev. 2 to Rev. 3 Changes

Chapter	Description
Throughout	Formatting, layout, spelling, and grammar corrections. Added information about the MCF51QE32 device. Changed the SRAM size for the MCF51QE64 device (was 4 Kbytes, is 8 Kbytes). Removed the "Preliminary" label from the footer.
Device Overview	Corrected the number of ADC channels for the MCF51QE64 device (was 22, is 20). Corrected the number of ADC channels for the 64-pin package of the MCF51QE64 device (was 22, is 20).
ColdFire Core	Corrected the reset value for the D1 register (was different for QE128 and QE64, is 0x1090_0050).
Analog Comparator	Updated the chapter contents to include the correct information about the low-power ACMP.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

MCF51QE128RM
Rev. 3, 09/2007

freescale™
semiconductor